Effective Inter-Procedural Static Analysis for the Linux Kernel

白家驹

清华大学计算机系

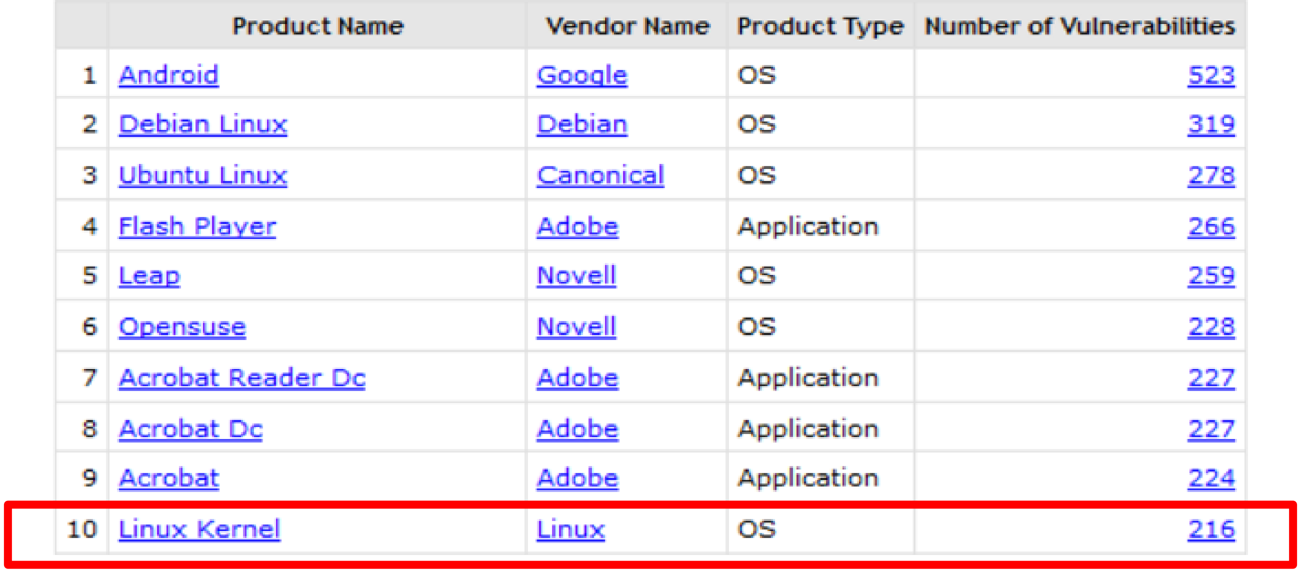
Tsinghua University

About Me

- **清华大学博士后（助理研究员）**
- **研究方向**
  - 操作系统可靠性
  - 内核程序分析
- **代表性工作**
  - 工具：基于动态分析或静态分析的内核缺陷检测
  - 论文：USENIX ATC 2019, 2018, 2016、ASPLOS 2019、ISSRE 2019、SANER 2019、CGO 2018、JSS 2018、……
- **联系方式**
  - 个人主页：https://baijiaju.github.io/
  - 电子邮箱：baijiaju1990@163.com

Background

- The Linux kernel is not reliable and safe as expected
  - In 2017, >2000 new real bugs are reported in the Linux kernel
  - In 2016, 216 new vulnerabilities are reported in the Linux kernel



	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Android	Google	OS	523
2	Debian Linux	Debian	OS	319
3	Ubuntu Linux	Canonical	OS	278
4	Flash Player	Adobe	Application	266
5	Leap	Novell	OS	259
6	Opensuse	Novell	OS	228
7	Acrobat Reader Dc	Adobe	Application	227
8	Acrobat Dc	Adobe	Application	227
9	Acrobat	Adobe	Application	224
10	Linux Kernel	Linux	OS	216

Background

- Static analysis can conveniently detect bugs
  - Without actually running the checked program
  - High code coverage
  - Easy to use and extend
  - ……
- Inter-procedural analysis can find many deep bugs involving function calls

Existing Static Approaches

- Cppcheck [1]
  - Integrated with many popular development tools
  - Detect bugs in C/C++ code

```
[drivers/gpu/drm/omapdrm/dss/output.c:127]: (error) Uninitialized variable: out
[drivers/gpu/drm/omapdrm/dss/output.c:148]: (error) Uninitialized variable: out
[drivers/gpu/drm/omapdrm/omap_debugfs.c:65]: (error) Uninitialized variable: fb
[drivers/gpu/drm/omapdrm/omap_dmm_tiler.c:215]: (warning) Possible null pointer dereference: engine
[drivers/gpu/drm/omapdrm/omap_dmm_tiler.c:218]: (warning) Possible null pointer dereference: engine
[drivers/gpu/drm/omapdrm/omap_dmm_tiler.c:219]: (warning) Possible null pointer dereference: engine
[drivers/gpu/drm/omapdrm/omap_dmm_tiler.c:532]: (error) Shifting signed 32-bit value by 31 bits is undefined behaviour
```

[1] Cppcheck: a tool for static C/C++ code analysis. http://cppcheck.sourceforge.net/

Existing Static Approaches

- Sparse [2]
  - Written by Linus Torvalds, maintained by Josh Triplett and Chris Li
  - Enabled when compiling the Linux kernel code

```
fs/gfs2/glock.c:1881:13: warning: context imbalance in 'gfs2_glock_seq_start' - wrong count at exit
include/linux/rcupdate.h:901:9: warning: context imbalance in 'gfs2_glock_seq_stop' - unexpected unlock
sound/oss/pas2_pcm.c:42:17: warning: symbol 'pas_audiodev' was not declared. Should it be static?
drivers/ata/libata-scsi.c:1915:9: warning: context imbalance in 'ata_scsi_rbuf_get' - wrong count at exit
drivers/ata/libata-scsi.c:1936:31: warning: context imbalance in 'ata_scsi_rbuf_fill' - unexpected unlock
drivers/ata/libata-scsi.c:1934:48: warning: context imbalance in 'atapi_qc_complete' - unexpected unlock
sound/oss/pas2_card.c:38:17: warning: symbol 'pas_translate_code' was not declared. Should it be static?
```

[2] Sparse: a static tool for the Linux kernel. https://elinux.org/Sparse/

Existing Static Approaches

- Coccinelle [3]
  - Developed by Julia Lawall and other people in LIP6
  - User can write rules in semantic patches to detect bugs

```
drivers/infiniband/core/uverbs_cmd.c:1530:1-3: WARNING: PTR_ERR_OR_ZERO can be used
drivers/virtio/virtio_mmio.c:666:1-3: WARNING: PTR_ERR_OR_ZERO can be used
drivers/clk/sunxi/clk-sun9i-mmc.c:108:26-29: ERROR: Missing resource_size with r
drivers/platform/x86/apple-gmux.c:464:25-28: WARNING: Suspicious code. resource_size is maybe missing with res
drivers/net/ethernet/ti/netcp_core.c:2012:2-7: WARNING: invalid free of devm_ allocated data
drivers/gpu/drm/tegra/sor.c:599:11-29: WARNING opportunity for simple_open, see also structure on line 673
drivers/video/fbdev/mbx/mbxdebugfs.c:18:11-28: WARNING opportunity for simple_open, see also structure on line 184
```

[3] Coccinelle: a program matching and transformation engine. http://coccinelle.lip6.fr/

Existing Static Approaches

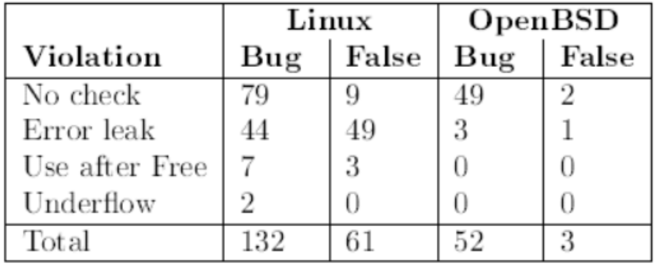
- Smatch [4]
  - Developed by Dan Carpenter
  - Has found over 3000 bugs in the Linux kernel

```
drivers/thermal/thermal_core.c:1452 thermal_generate_netlink_event() warn: can 'thermal_event' even be NULL?
drivers/tty/serial/serial_core.c:288 uart_shutdown() error: we previously assumed 'uport' could be null (see line 284)
drivers/tty/vt/vt.c:3347 con_init() error: potential null dereference 'vc'.  (kzalloc returns null)
drivers/tty/synclinkmp.c:729 install() error: we previously assumed 'info' could be null (see line 725)
drivers/scsi/sg.c:489 sg_read() error: we previously assumed 'srp' could be null (see line 468)
drivers/scsi/fcoe/fcoe.c:2243 _fcoe_create() error: potential null dereference 'fcoe'.  (fcoe_interface_create returns null)
drivers/md/dm-log-userspace-transfer.c:110 fill_pkg() error: we previously assumed 'tfr' could be null (see line 84)
```

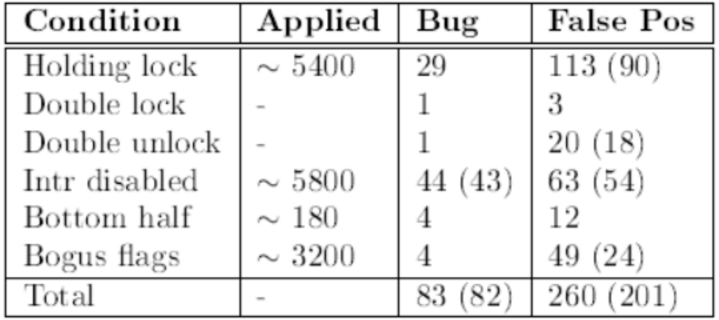
[4] Smatch: a static bug-finding tool for the Linux kernel. http://smatch.sourceforge.net/.

Existing Static Approaches

- XGCC [5]
  - Developed by the team of Dawson Engler
  - The original tool of Coverity [6]



	Linux		OpenBSD	
Violation	Bug	False	Bug	False
No check	79	9	49	2
Error leak	44	49	3	1
Use after Free	7	3	0	0
Underflow	2	0	0	0
Total	132	61	52	3

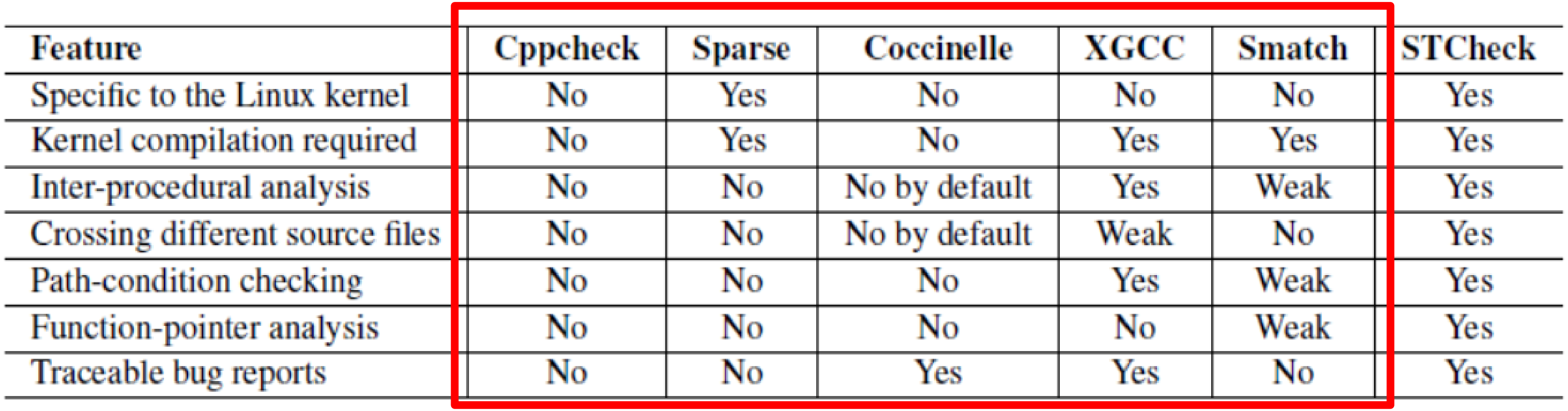


Condition	Applied	Bug	False Pos
Holding lock	~ 5400	29	113 (90)
Double lock	-	1	3
Double unlock	-	1	20 (18)
Intr disabled	~ 5800	44 (43)	63 (54)
Bottom half	~ 180	4	12
Bogus flags	~ 3200	4	49 (24)
Total	-	83 (82)	260 (201)

[5] Dawson Engler, et. al. Checking system rules using system-specific, programmer-written compiler extensions. In OSDI 2000.

[6] Coverity: a commercial static analysis tool. https://scan.coverity.com.

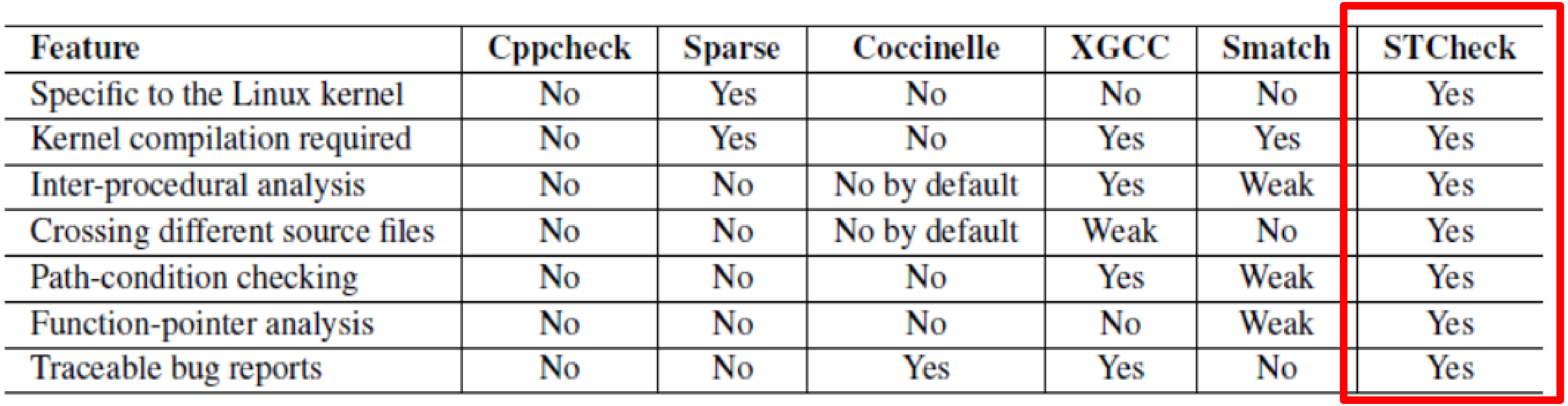
Existing Static Approaches



Feature	Cppcheck	Sparse	Coccinelle	XGCC	Smatch	STCheck
Specific to the Linux kernel	No	Yes	No	No	No	Yes
Kernel compilation required	No	Yes	No	Yes	Yes	Yes
Inter-procedural analysis	No	No	No by default	Yes	Weak	Yes
Crossing different source files	No	No	No by default	Weak	No	Yes
Path-condition checking	No	No	No	Yes	Weak	Yes
Function-pointer analysis	No	No	No	No	Weak	Yes
Traceable bug reports	No	No	Yes	Yes	No	Yes

- Common limitations
  - Inter-procedural analysis
  - Function-pointer analysis
  - Cross-source-file analysis
  - ……

Goal



Feature	Cppcheck	Sparse	Coccinelle	XGCC	Smatch	STCheck
Specific to the Linux kernel	No	Yes	No	No	No	Yes
Kernel compilation required	No	Yes	No	Yes	Yes	Yes
Inter-procedural analysis	No	No	No by default	Yes	Weak	Yes
Crossing different source files	No	No	No by default	Weak	No	Yes
Path-condition checking	No	No	No	Yes	Weak	Yes
Function-pointer analysis	No	No	No	No	Weak	Yes
Traceable bug reports	No	No	Yes	Yes	No	Yes

- STCheck
  - Effective inter-procedural analysis
  - Accurate function-pointer analysis
  - Precise cross-source-file analysis
  - ……

Challenges

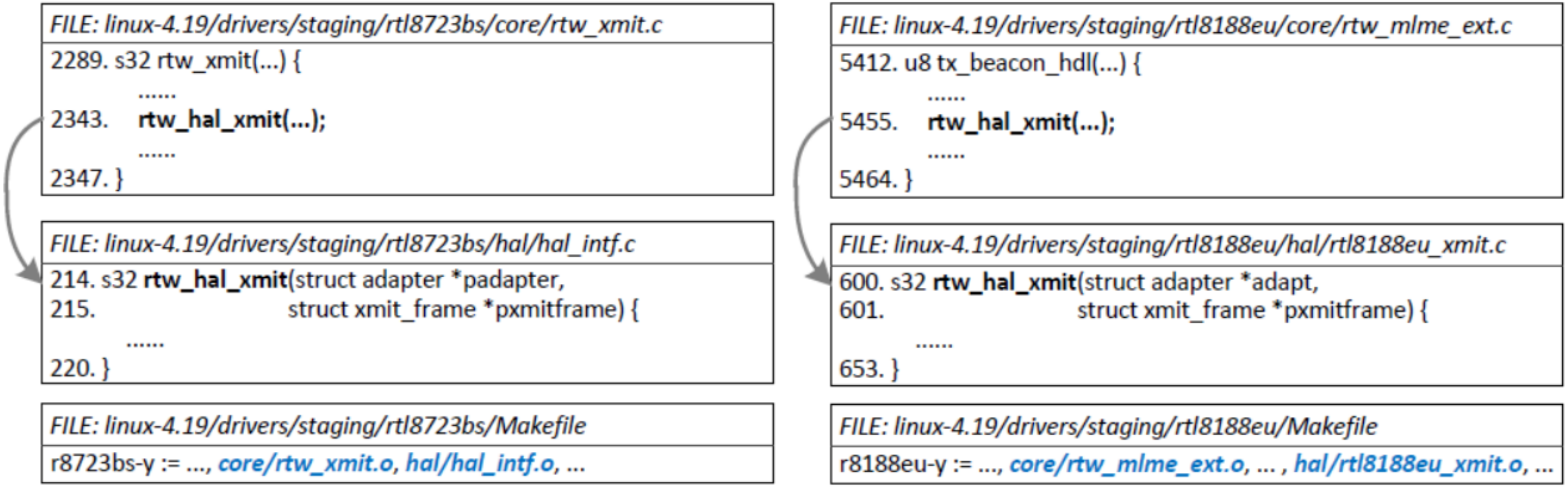
- Handling called functions
  - There are lots of function calls across different source files
  - Many functions share their names with some other functions
- Function-pointer analysis
  - There are lots of function-pointer calls
  - Incomplete call graphs or incorrect call graphs
- Linux kernel code base is very large and complex
  - Long analysis time
  - Much memory cost

Key Techniques in STCheck

- Handling called functions
  - Staged strategy
  - First collect useful information and then perform static analysis
- Function-pointer analysis
  - Connection-based alias analysis
  - Identify connections between calls and candidate referenced functions
- Linux kernel code base is very large and complex
  - Adaptive summary-based analysis
  - Use deduplication, lifetime management and reference counting

Staged Strategy

- Link connection
  - Collect source files that are linked to generate the same module
  - Intercept the link procedural during code compilation



```
FILE: linux-4.19/drivers/staging/rtl8723bs/core/rtw_xmit.c
2289. s32 rtw_xmit(...) {
          ......
2343.     rtw_hal_xmit(...);
          ......
2347. }
```

```
FILE: linux-4.19/drivers/staging/rtl8723bs/hal/hal_intf.c
214. s32 rtw_hal_xmit(struct adapter *padapter,
215.                  struct xmit_frame *pxmitframe) {
         ......
220. }
```

```
FILE: linux-4.19/drivers/staging/rtl8723bs/Makefile
r8723bs-y := ..., core/rtw_xmit.o, hal/hal_intf.o, ...
```

(a) rtl8723bs

```
FILE: linux-4.19/drivers/staging/rtl8188eu/core/rtw_mlme_ext.c
5412. u8 tx_beacon_hdl(...) {
          ......
5455.     rtw_hal_xmit(...);
          ......
5464. }
```

```
FILE: linux-4.19/drivers/staging/rtl8188eu/hal/rtl8188eu_xmit.c
600. s32 rtw_hal_xmit(struct adapter *adapt,
601.                  struct xmit_frame *pxmitframe) {
         ......
653. }
```

```
FILE: linux-4.19/drivers/staging/rtl8188eu/Makefile
r8188eu-y := ..., core/rtw_mlme_ext.o, ... , hal/rtl8188eu_xmit.o, ...
```

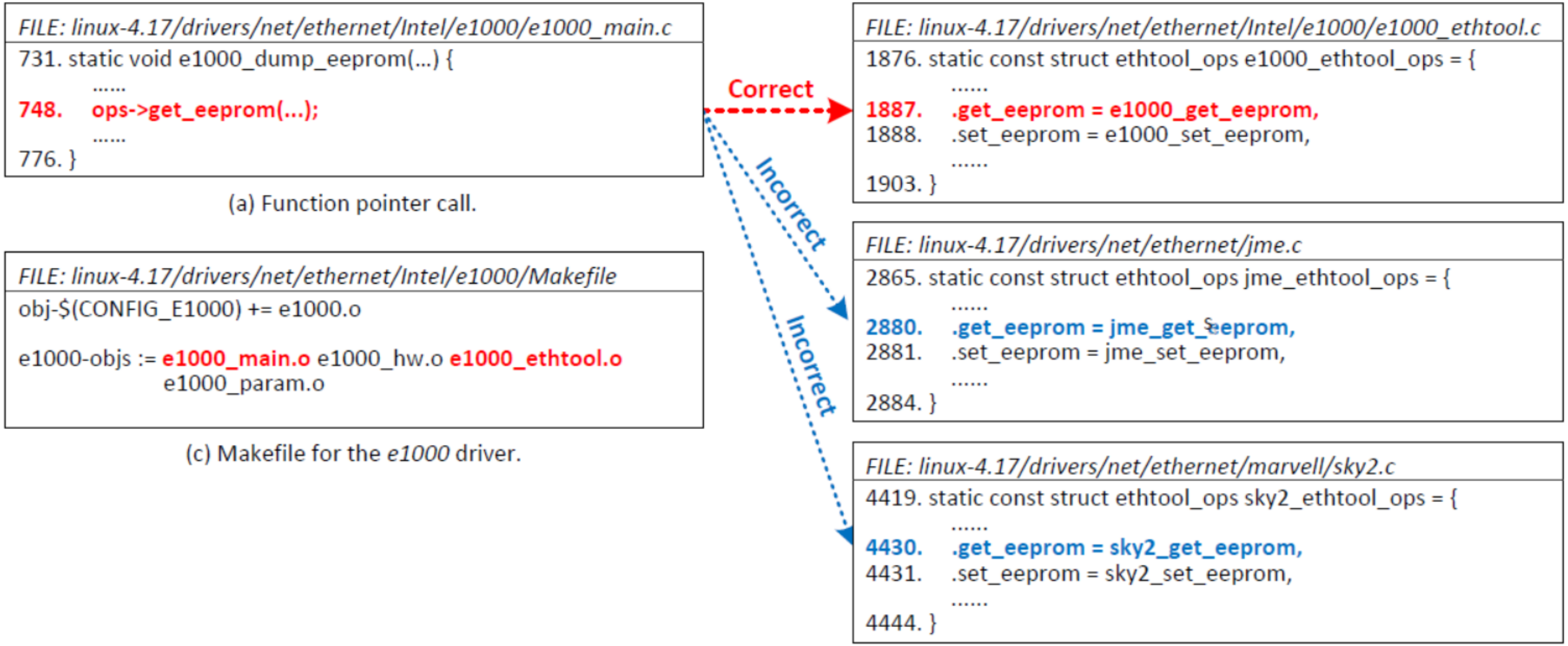
(b) rtl8188eu

Connection-Based Alias Analysis

- S1: Handle function-pointer assignments
- S2: Identify candidate referenced functions according to data structure field and function type
- S3: Check the connection between the source files of function-pointer call and candidate referenced functions

Connection-Based Alias Analysis

- Link connection



```
FILE: linux-4.17/drivers/net/ethernet/Intel/e1000/e1000_main.c
731. static void e1000_dump_eeprom(...) {
         ......
748.     ops->get_eeprom(...);
         ......
776. }
```

(a) Function pointer call.

```
FILE: linux-4.17/drivers/net/ethernet/Intel/e1000/Makefile
obj-$(CONFIG_E1000) += e1000.o

e1000-objs := e1000_main.o e1000_hw.o e1000_ethtool.o
              e1000_param.o
```

(c) Makefile for the *e1000* driver.

```
FILE: linux-4.17/drivers/net/ethernet/Intel/e1000/e1000_ethtool.c
1876. static const struct ethtool_ops e1000_ethtool_ops = {
          ......
1887.     .get_eeprom = e1000_get_eeprom,
1888.     .set_eeprom = e1000_set_eeprom,
          ......
1903. }
```

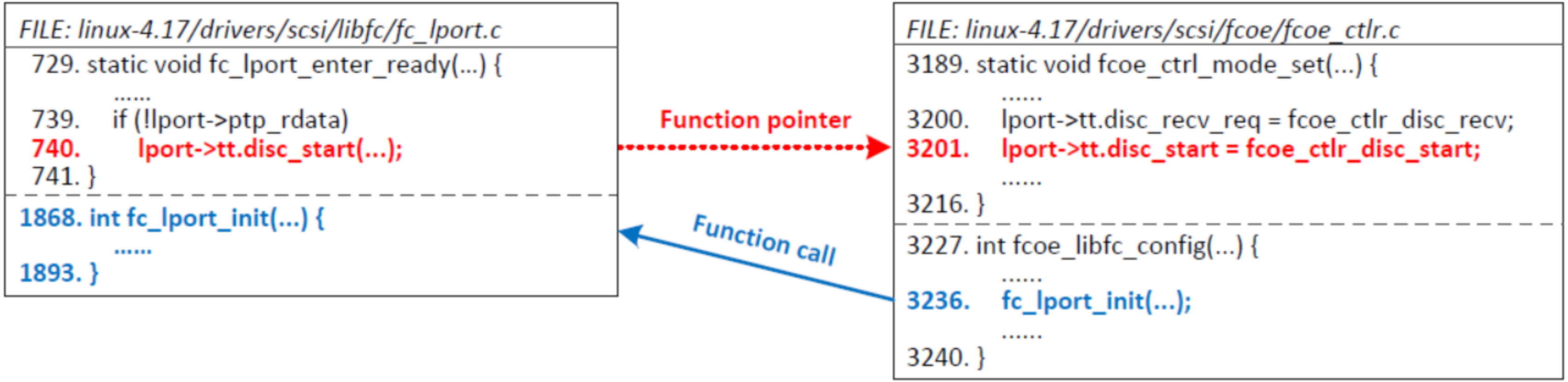
```
FILE: linux-4.17/drivers/net/ethernet/jme.c
2865. static const struct ethtool_ops jme_ethtool_ops = {
          ......
2880.     .get_eeprom = jme_get_eeprom,
2881.     .set_eeprom = jme_set_eeprom,
          ......
2884. }
```

```
FILE: linux-4.17/drivers/net/ethernet/marvell/sky2.c
4419. static const struct ethtool_ops sky2_ethtool_ops = {
          ......
4430.     .get_eeprom = sky2_get_eeprom,
4431.     .set_eeprom = sky2_set_eeprom,
          ......
4444. }
```

(b) Some functions that may be referenced by the function pointer.

Connection-Based Alias Analysis

- Function-call connection (direct)

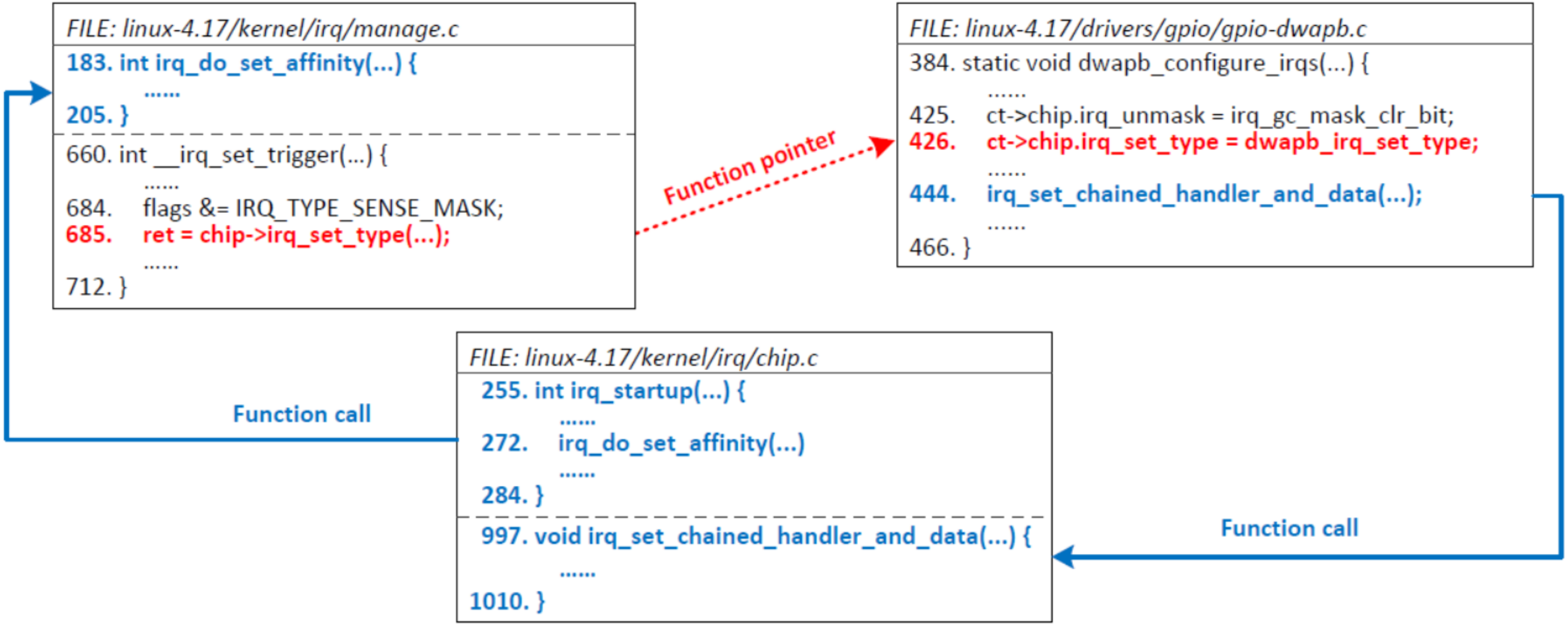


```
FILE: linux-4.17/drivers/scsi/libfc/fc_lport.c
 729. static void fc_lport_enter_ready(...) {
          ......
 739.     if (!lport->ptp_rdata)
 740.         lport->tt.disc_start(...);
 741. }
- - - - - - - - - - - - - - - - - - - - - - - - -
1868. int fc_lport_init(...) {
          ......
1893. }
```

```
FILE: linux-4.17/drivers/scsi/fcoe/fcoe_ctlr.c
3189. static void fcoe_ctrl_mode_set(...) {
          ......
3200.     lport->tt.disc_recv_req = fcoe_ctlr_disc_recv;
3201.     lport->tt.disc_start = fcoe_ctlr_disc_start;
          ......
3216. }
- - - - - - - - - - - - - - - - - - - - - - - - -
3227. int fcoe_libfc_config(...) {
          ......
3236.     fc_lport_init(...);
          ......
3240. }
```

Connection-Based Alias Analysis

- Function-call connection (indirect)



```
FILE: linux-4.17/kernel/irq/manage.c
183. int irq_do_set_affinity(...) {
         ......
205. }
- - - - - - - - - - - - - - - - - - - - -
660. int __irq_set_trigger(...) {
         ......
684.     flags &= IRQ_TYPE_SENSE_MASK;
685.     ret = chip->irq_set_type(...);
         ......
712. }
```

```
FILE: linux-4.17/drivers/gpio/gpio-dwapb.c
384. static void dwapb_configure_irqs(...) {
         ......
425.     ct->chip.irq_unmask = irq_gc_mask_clr_bit;
426.     ct->chip.irq_set_type = dwapb_irq_set_type;
         ......
444.     irq_set_chained_handler_and_data(...);
         ......
466. }
```

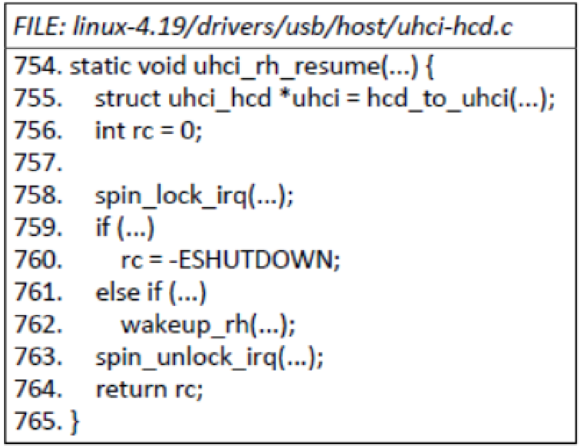
```
FILE: linux-4.17/kernel/irq/chip.c
 255. int irq_startup(...) {
          ......
 272.     irq_do_set_affinity(...)
          ......
 284. }
- - - - - - - - - - - - - - - - - - - - -
 997. void irq_set_chained_handler_and_data(...) {
          ......
1010. }
```

Adaptive Summary-Based Analysis

- Summary-based analysis
  - Store the results of previous analyses as *summaries*, and reuse them to avoid repeated analyses
  - A function summary often contains the information of each code path
  - Storing summaries often require much memory for large code bases
  - How to reduce memory cost?

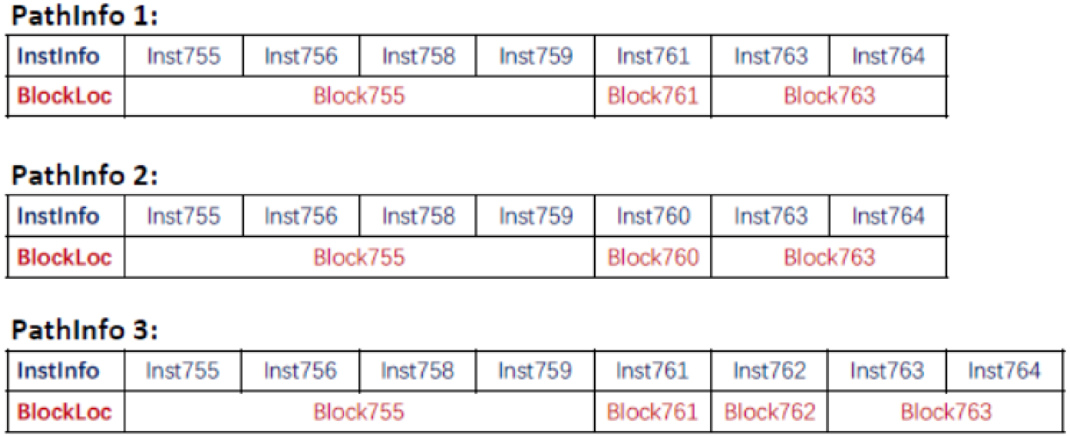
Adaptive Summary-Based Analysis

- Deduplication



FILE: linux-4.19/drivers/usb/host/uhci-hcd.c

```
754. static void uhci_rh_resume(...) {
755.     struct uhci_hcd *uhci = hcd_to_uhci(...);
756.     int rc = 0;
757.
758.     spin_lock_irq(...);
759.     if (...)
760.         rc = -ESHUTDOWN;
761.     else if (...)
762.         wakeup_rh(...);
763.     spin_unlock_irq(...);
764.     return rc;
765. }
```



PathInfo 1:

InstInfo	Inst755	Inst756	Inst758	Inst759	Inst761	Inst763	Inst764
BlockLoc	Block755				Block761	Block763	

PathInfo 2:

InstInfo	Inst755	Inst756	Inst758	Inst759	Inst760	Inst763	Inst764
BlockLoc	Block755				Block760	Block763	

PathInfo 3:

InstInfo	Inst755	Inst756	Inst758	Inst759	Inst761	Inst762	Inst763	Inst764
BlockLoc	Block755				Block761	Block762	Block763	

Duplicated InstInfo: Inst755(3), Inst756(3), Inst758(3), Inst 759(3), Inst761(2), Inst763(3), Inst764(3)

Duplicated BlockLoc: Block755(3), Block761(2), Block763(3)

Adaptive Summary-Based Analysis

- Lifetime management
  - Some function summaries may never be used during static analysis
  - Maintaining summary's lifetime to release it when its lifetime ends
- Some details
  - Function definition

    internal, external, exported
  - Call times of a function

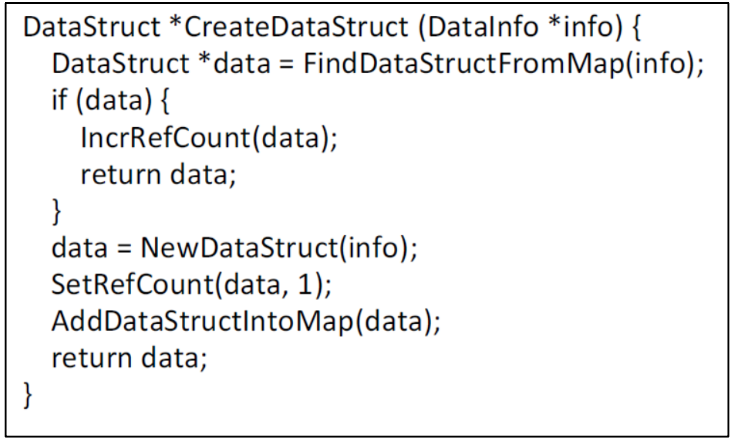
    call_time = *total_time*?
  - Order of analyzing source files

    MyFunc is defined in *S1.c* and called in *S3.c*

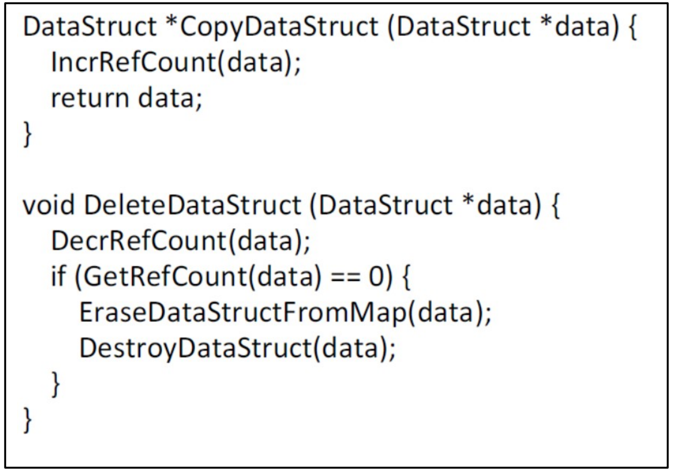
    <*S1.c, S2.c, S3.c*> VS. <*S1.c, S3.c, S2.c*>

Adaptive Summary-Based Analysis

- Reference counting
  - Safely release a summary
  - Add a reference counter in each shared data item



```
DataStruct *CreateDataStruct (DataInfo *info) {
   DataStruct *data = FindDataStructFromMap(info);
   if (data) {
      IncrRefCount(data);
      return data;
   }
   data = NewDataStruct(info);
   SetRefCount(data, 1);
   AddDataStructIntoMap(data);
   return data;
}
```

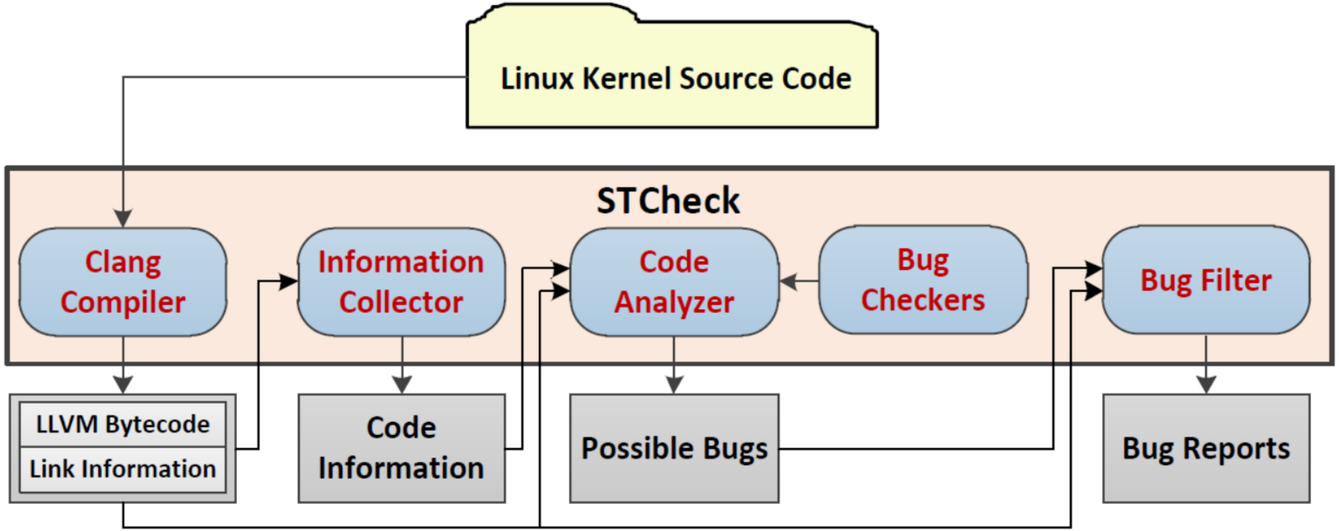


```
DataStruct *CopyDataStruct (DataStruct *data) {
   IncrRefCount(data);
   return data;
}

void DeleteDataStruct (DataStruct *data) {
   DecrRefCount(data);
   if (GetRefCount(data) == 0) {
      EraseDataStructFromMap(data);
      DestroyDataStruct(data);
   }
}
```

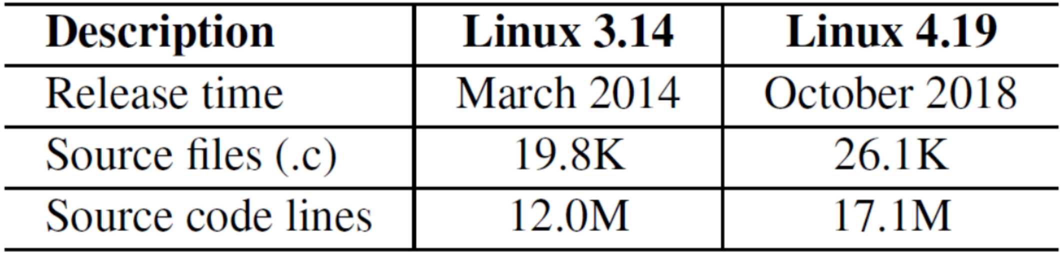
Our Approach

- STCheck
  - Implemented based on LLVM
  - Flow-sensitive, context-sensitive, field-sensitive, inter-procedural
  - Support independent bug checkers to detect different kinds of bugs



Evaluation

- Linux 3.14 and 4.19
  - Use a common PC with four CPUs and 16GB memory
  - Run on four threads
  - Make *allyesconfig* of x86
  - Detect null-pointer dereferences and double-lock/unlock bugs



Description	Linux 3.14	Linux 4.19
Release time	March 2014	October 2018
Source files (.c)	19.8K	26.1K
Source code lines	12.0M	17.1M

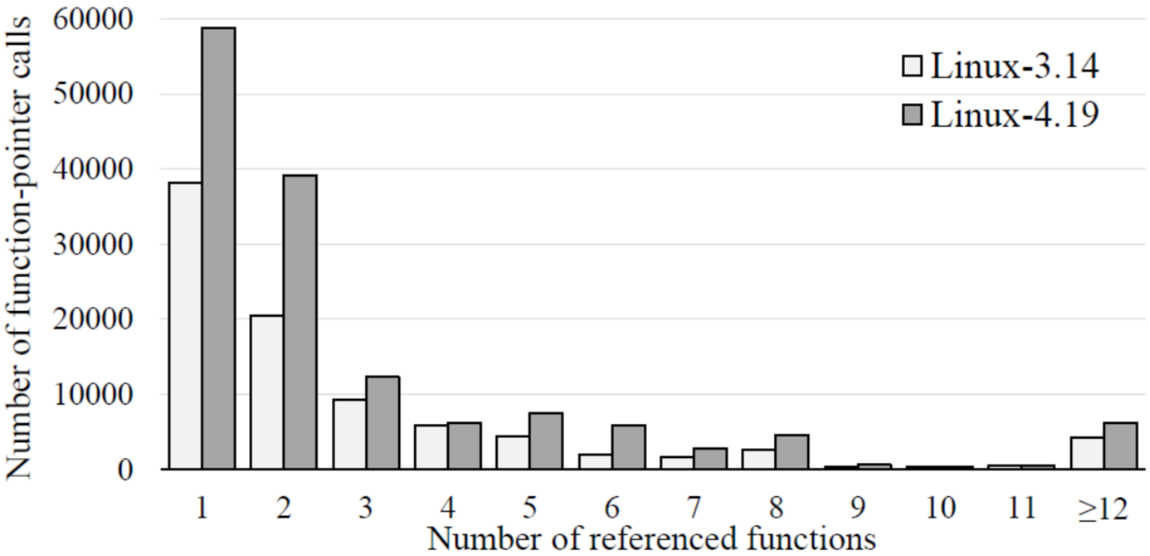
Evaluation

- Code analysis

Description		Linux 3.14	Linux 4.19
Code handling	Handled source files (.c)	11.3K	17.1K
	Handled source code lines	8.5M	12.4M
	Handled functions	203K	413K
Function-pointer analysis	Encountered function-pointer calls	131K	187K
	Handled function-pointer calls	94K	145K
	Identified referenced functions	264K	459K
Time usage		449m	580m
Memory cost		8.3GB	10.6GB

Evaluation

- Function-pointer analysis
  - Most of function-pointer calls have only 1-2 referenced functions



Evaluation

- Bug detection
  - 127 found bugs in Linux 3.14 has been fixed in Linux 4.19
  - 182 of 250 reported bugs in Linux 4.19 have been confirmed

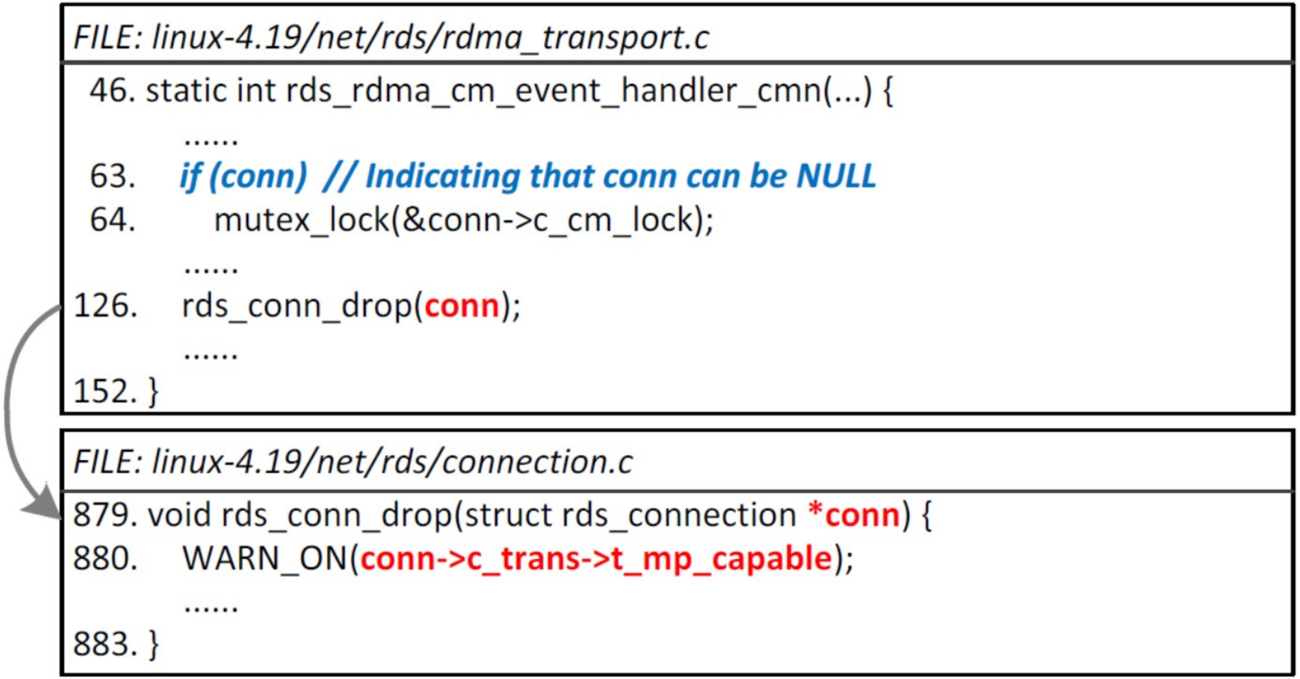
Description	**Linux 3.14**	**Linux 4.19**
Null-pointer deferences (real / all)	416 / 535	732 / 919
Double-lock/unlock bugs (real / all)	13 / 19	12 / 17
Total (real / all)	429 / 554	744 / 936

Some confirmed bugs:

- https://github.com/torvalds/linux/commit/f2538f999345
- https://github.com/torvalds/linux/commit/0e7bf23e4967
- https://github.com/torvalds/linux/commit/627469e4445

Evaluation

- Example null-pointer dereference



FILE: linux-4.19/net/rds/rdma_transport.c

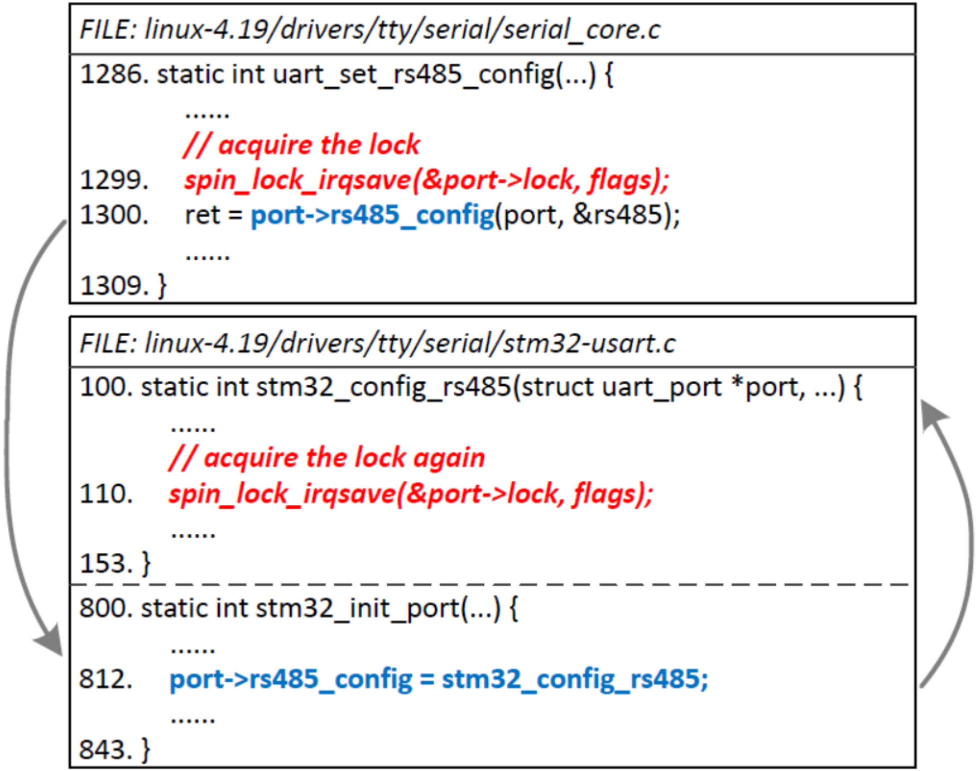
```
static int rds_rdma_cm_event_handler_cmn(...) {
    ......
    if (conn)  // Indicating that conn can be NULL
        mutex_lock(&conn->c_cm_lock);
    ......
    rds_conn_drop(conn);
    ......
}
```

FILE: linux-4.19/net/rds/connection.c

```
void rds_conn_drop(struct rds_connection *conn) {
    WARN_ON(conn->c_trans->t_mp_capable);
    ......
}
```

Evaluation

- Example double-lock bug

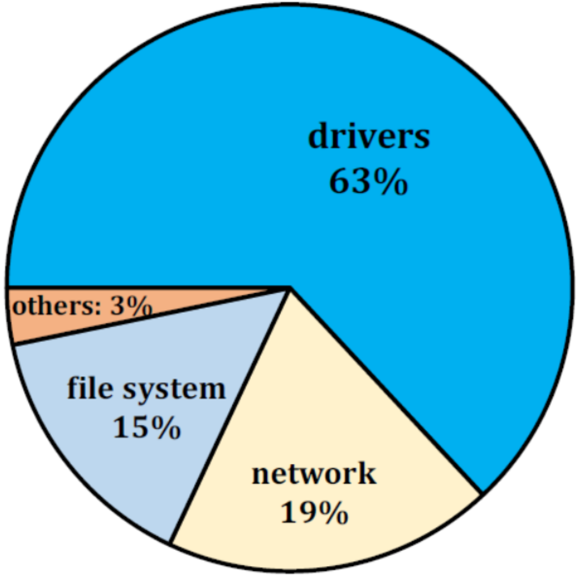


```
FILE: linux-4.19/drivers/tty/serial/serial_core.c
1286. static int uart_set_rs485_config(...) {
          ......
          // acquire the lock
1299.     spin_lock_irqsave(&port->lock, flags);
1300.     ret = port->rs485_config(port, &rs485);
          ......
1309. }
```

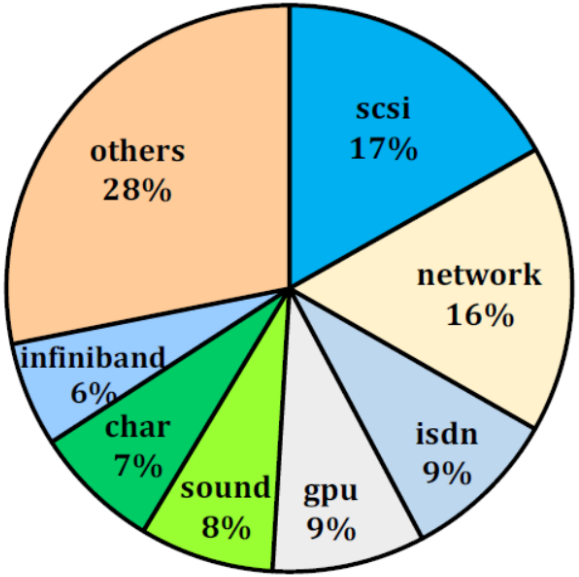
```
FILE: linux-4.19/drivers/tty/serial/stm32-usart.c
100. static int stm32_config_rs485(struct uart_port *port, ...) {
         ......
         // acquire the lock again
110.     spin_lock_irqsave(&port->lock, flags);
         ......
153. }
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
800. static int stm32_init_port(...) {
         ......
812.     port->rs485_config = stm32_config_rs485;
         ......
843. }
```

Evaluation

- Bug distribution



(a) Whole kernel



(b) Drivers

Comparison to Existing Approaches

- Checked kernel: *Linux 4.19*
- Bug type: *null-pointer dereference*
- Target tools: *Cppcheck and Smatch*

Comparison to Existing Approaches

- Cppcheck
  - No effective inter-procedural analysis
  - No function-pointer analysis
  - No need of kernel configuration
- Results
  - 35 null-pointer dereferences, and 9 are real
  - Cppcheck finds 9 real bugs missed by STCheck
  - ***STCheck finds 732 real bugs missed by Cppcheck***
  - STCheck achieves lower false positive rate

Comparison to Existing Approaches

- Smatch
  - Simple inter-procedural analysis
  - Simple function-pointer analysis
  - No cross-source-file analysis
- Results
  - 362 null-pointer dereferences, and 105 are real
  - Smatch and STCheck find 65 identical real bugs
  - Smatch finds 40 real bugs missed by STCheck
  - ***STCheck finds 667 bugs missed by Smatch***
  - STCheck achieves lower false positive rate

Limitations

- False positives
  - Fail to identify feasible code paths in some complex cases
  - Some implemented low-level analyses (such as pointer analysis) may be inaccurate
  - Function-pointer analysis may identify wrong referenced functions
  - ……
- False negatives
  - The length of analyzed call chains is limited
  - Function-pointer analysis cannot handle function-pointer assignments in some complex cases
  - ……

Conclusion

- Inter-procedural analysis of the Linux kernel is hard
- STCheck: automated and effective
  - Staged strategy
  - Connection-based function-pointer analysis
  - Adaptive summary-based analysis
- Find hundreds of new real bugs in the Linux kernel

Something else

- Our previous works of static kernel-code analysis
  - DSAC: detect sleep-in-atomic-context bugs
  - DCNS: detect conservative non-sleep defects
  - DCUAF: detect concurrency use-after-free bugs
  - ……
- Our previous works of dynamic kernel-code analysis
  - EH-Test: Single fault-injection testing
  - DILP: detect data races caused by inconsistent lock protection
  - FIZZER: fault-injection-based fuzzing
  - ……

We are looking forward to checking real industry system code!

Thanks!

Q & A

Email: baijiaju1990@163.com

Homepage: https://baijiaju.github.io/