基于接口规则的设备驱动可靠性研究

刘虎球

清华大学·计算机系

liuhq11@mails.tsinghua.edu.cn

个人简介

- 刘虎球(Hu-Qiu Liu)
  - 提前攻博二年级
  - 导师：胡事民教授
  - 邮箱: *huqiuliu@gmail.com*
    liuhq11@mails.tsinghua.edu.cn
  - 电话: +86 134 8871 8103
  - 主要研究的是内核扩展(驱动)与内核直接的接口使用规则，提升驱动可靠性
  - 日常从事操作系统的调试和开发工作

Agenda

一. 读博动机(3分钟)

二. 研究内容介绍（50分钟）

- 一. Idea起源
- 二. 研究框架
- 三. PF-Miner

三. 研究感悟（2分钟）

四. 实验演示（15分钟）

五. Q&A

一、读博动机

- 天时、地利、人和
  - 动力、平台、自信、努力
    - 树立远大的学术理想
    - 好的导师平台（课题、积累、政策、出国交流）
    - 时间+耐心、有计划推进
    - 百倍的努力
      - 文献多读、泛读精读相结合
      - 实验多试、多看源码、多看日志
      - 养成一些良好的习惯：笔记、思考、乐观、坚持、交流
  - 依据实力投稿，早日毕业
    - 为毕业而发文章

一、读博动机

- 天时、地利、人和
  - 动力：按期毕业、出国开会、不做研究者
  - 平台：时间自由、导师放养
  - 自信：发差点的文章?
  - 努力：记录文献、标记文献、计划推进
- 研究成果
  - 计算机学报 2篇
  - 国际会议 3篇(欧洲、韩国、美国)
  - 二作、三作若干篇

Agenda

一．读博动机

二．研究内容介绍

- 一．Idea起源
- 二．研究框架
- 三．PF-Miner

三．研究感悟

四．实验演示

五．Q&A

Agenda

一．读博动机

二．研究内容介绍

一．Idea起源

二．研究框架

三．PF-Miner

三．研究感悟

四．实验演示

二、研究简介-Idea起源

- 源自工程开发
  - 帧寄存器末位对齐、DMA地址低位对齐

```
177.    static void configure_hc(struct uhci_hcd *uhci)
……      {
……       ……
184.     /* Store the frame list base address */
185.     outl(uhci->frame_dma_handle, uhci->io_addr
        + USBFLBASEADD);
……       ……
201.    }
```

二、研究简介-Idea起源

- 源自工程开发
  - 帧寄存器末位对齐、DMA地址低位对齐

表1 帧基址寄存器

表项	内容	备注
名称	FLBASEADD	帧基址寄存器
地址	基地址+(08-0B)H	
属性	可读可写	
宽度	32比特	11:0被保留

怎么办?

开发者：修复bug继续开发?
程序控：如何帮你定位?
研究者：自动检查规则?

二、研究简介-Idea起源

- 源自工程开发
  - 帧寄存器末位对齐、DMA地址低位对齐
    - 有益之处：快速定位升级引起的配置问题
    - 抽象规则：简单定义对齐的模板·找类似问题
    - 插装点：待测目标
    - 自动修复?
      - 加assert断言
      - 自动记录异常日志?
    - 整体拔高
      - 理论分析？算法建模？完整的实验测评（理论分析、动态性能、对比版本之间的评测、不同方法对比）？
  - 《计算机学报》 2014第10期

Agenda

一．读博动机

二．研究内容介绍

- 一．Idea起源
- 二．研究框架
- 三．PF-Miner

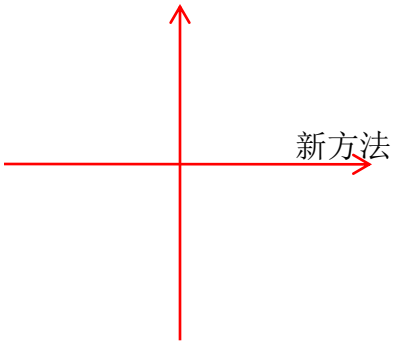
三．研究感悟

四．实验演示

五．Q&A

二、研究简介-研究框架

- 研究的关联与深化
  - 当执行出错时，驱动模块应终止运行
    - 调用接口函数释放资源（内存·锁·操作回滚）
    - 问题：接口函数怎么来?
      - 调研：有没有类似的？有没有问题？自己找?
      - 去哪找：找别人没找过的场景
        - 场景等价于异常路径（PF-Miner）、驱动结构的跨函数之间(Pair-Miner)、二进制上(bMiner)
      - 怎么找：用实用的+理论型方法找
        - DM的方法、NLP的方法、ML的方法
      - 找对没：用合适的方法进行评估测试
        - 对比分析、找人验证、手动验证等
    - 拔高：理论建模？算法？实验？



二、研究简介-研究框架

- 研究的系统化
  - 基于先验规则的设备驱动可靠性研究
    - 驱动配置规则(AiLsDc)
  - 跨函数的规则挖掘与检测
    - PairMiner
  - 面向异常处理的配对规则挖掘与检测
    - PF-Miner、bMiner
  - 驱动接口的顺序依赖规则挖掘与检测
    - SD-Miner

二、研究简介-研究框架

- 研究的系统化
  - 基于先验规则的设备驱动可靠性研究
    - 驱动配置规则(AiLsDc)
  - 跨函数的规则挖掘与检测
    - PairMiner
  - **面向异常处理的配对规则挖掘与检测**
    - PF-Miner、bMiner
  - 驱动接口的顺序依赖规则挖掘与检测
    - SD-Miner

Agenda

一．读博动机

二．研究内容介绍

一．Idea起源

二．研究框架

三．PF-Miner

三．研究感悟

四．实验演示

五．Q&A

PF-Miner: A new paired functions mining method for Android kernel in error paths

Liu Huqiu, Wang Yuping etc.

Tsinghua University

liuhq11@mails.tsinghua.edu.cn

Agenda

I . Background

II. Related works

III. implementation for PF-Miner

IV. Experimental evaluation

V. Conclusion

VI. References

Agenda

I . Background

II. Related works

III. implementation for PF-Miner

IV. Experimental evaluation

V. Conclusion

VI. References

I . Background

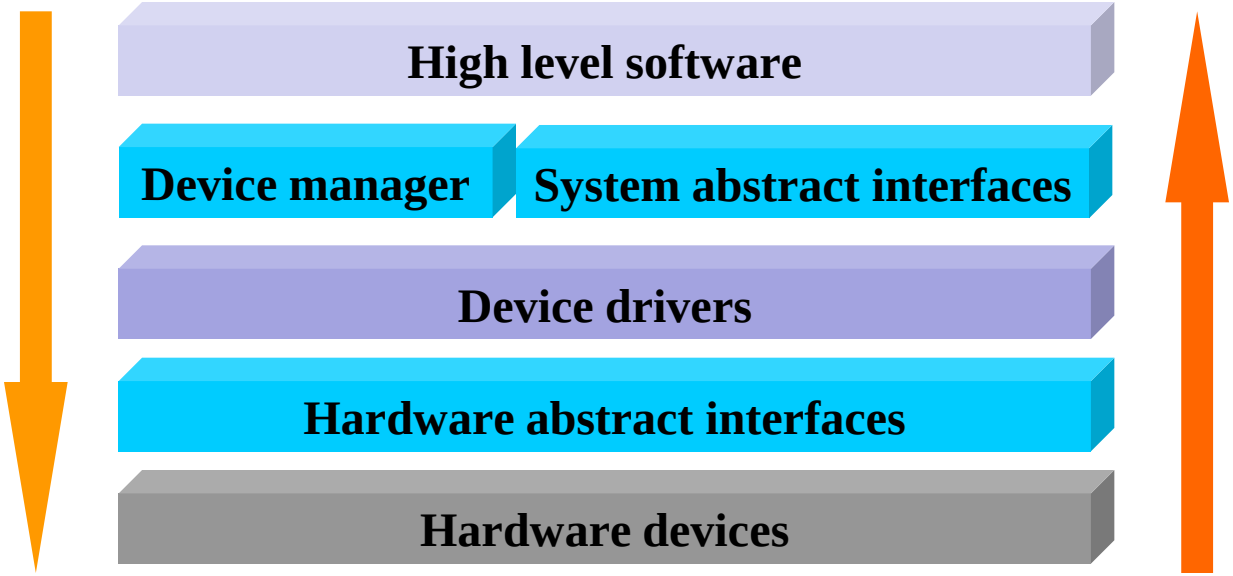
- About Android kernel
- Device drivers
- Implicit rules
- Rules extraction and violation detection

I . Background

- Android kernel
  - Separate from Linux kernel 2.6.39, and the current latest version is 3.10.10
  - Under Android framework
  - Special drivers for Android, and variety of Linux drivers are imported directly

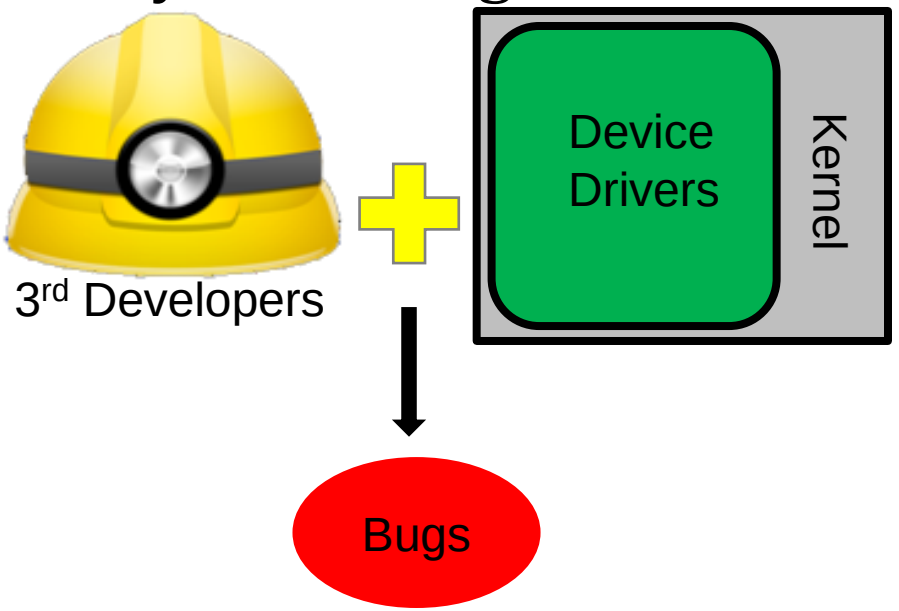
I . Background

- About device driver
  - Run in kernel mode with kernel extension modules
  - Uses kernel extension interfaces to communicate with kernels



I . Background

- About device driver
  - Developed by different company and org
  - Unaware or ignore the rules for interfaces
  - Lack documents for interfaces
  - Uses interfaces incorrectly and brings lots of bugs into drivers



I . Background

- What rules?
  - Drivers uses kernel extension function to manage resources, as fig.

501	static int au_ide_probe(struct platform_device *dev)	**Function Name**
.....	{	
531	if (!request_mem_region(...) {	Normal execution
532	ret = -EBUSY;	Error handling
533	goto out;	
.....	}	
537	ahwif->regbase = (u32)ioremap(...);	Normal execution
538	if (ahwif->regbase == 0) {	Error handling
539	ret = -ENOMEM;	Error handling
540	goto out;	Error handling
541	}	
.....	.....	
566	out:	
567	return ret;	Error handling
568	}	

I . Background

- What rules?
  - Drivers uses kernel extension function to manage resources, as fig.
  - Many of them should be used in pairs, such as request_mem_region / release_mem_region, which are named as **paired functions**
  - In error handling paths, the function should release the acquired resources

I . Background

- Why rules?
  - Developers only care the implementation of normal execution functions, and many bugs hide in error handling paths
  - Ignoring paired functions will cause many complicate problems, such as memory leaks
  - Influence the reliability of drivers. Bugs in drivers is 3-7 times than kernel, and 85% crashes are related to drivers.

I . Background

- Target for PF-Miner
  - Automatically extract paired function from the source code of drivers
  - Detect violations with extracted paired functions which are used in error handling paths
  - Reports potential bugs

Agenda

I . Background

II. Related works

III. Implementation for PF-Miner

IV. Experimental evaluation

V. Conclusion

VI. References

II . Related works

- Reliability for drivers
- Rules mining
  - Generic rules mining
  - Specific rules mining
- Structure inferring

II . Related works

- Reliability for drivers
  - Dynamic recovery
    - Nooks
    - Shadow Driver
    - FGFT
  - Type checking
    - KINT etc.
  - Symbolic execution
    - S2E and SymDrive
- Paired functions is unknown for these methods before mining.

II . Related works

- Rules mining
  - ECC
  - PR-Miner:
  - CAR-Miner:
  - WYSIWIB
  - WN-Miner

II . Related works

- Rules mining
  - ECC
    - <a> must be paired with <b>, ECC also checks the source code, and extracts rules from normal execution paths, such as spin_unlock to spin_lock
  - PR-Miner
    - It uses data mining technique to extract implicit rules from large software code
    - Extract longest execution path pattern with several functions and variables

II . Related works

- Rules mining
  - CAR-Miner
    - It mines exception-handling rules in the form of sequence association rules by studying the open source projects on the web, and then uses these rules to detect violations.
  - WN-Miner
    - A specification mining in exception handling paths to learn temporal safety rules
    - It helps programmers to locate mistakes in exception circumstances or among uncommon code paths

II . Related works

- Specific rules mining
  - Icomment
  - aComment

II . Related works

- Specific rules mining
  - Icomment
    - Learn from source code and comments in large software codes
    - Detect inconsistency between source code notation and program comments

II . Related works

- Specific rules mining
  - aComment: Detect bugs for interrupts related

drivers/ssb/pcmcia.c:

```
static void ssb_pcmcia_write16(…) {…
    spin_lock_irqsave(…);
    err = select_core_and_segment(…);
    …
}
```

@IRQ (X, D)
@IRQ (E, E)

linux//arch/x86/mm/pageattr.c:

```
static void /* @IRQ (E, E) */ cpa_flush_array(…)
{ … BUG_ON(irqs_disabled()); … }
```

II . Related works

- Specific rules mining
  - aComment

drivers/ssb/pcmcia.c:

```
static void ssb_pcmcia_write16(…) {…
    spin_lock_irqsave(…);
    err = select_core_and_segment(…);
    …
}
```

@IRQ (X, D)

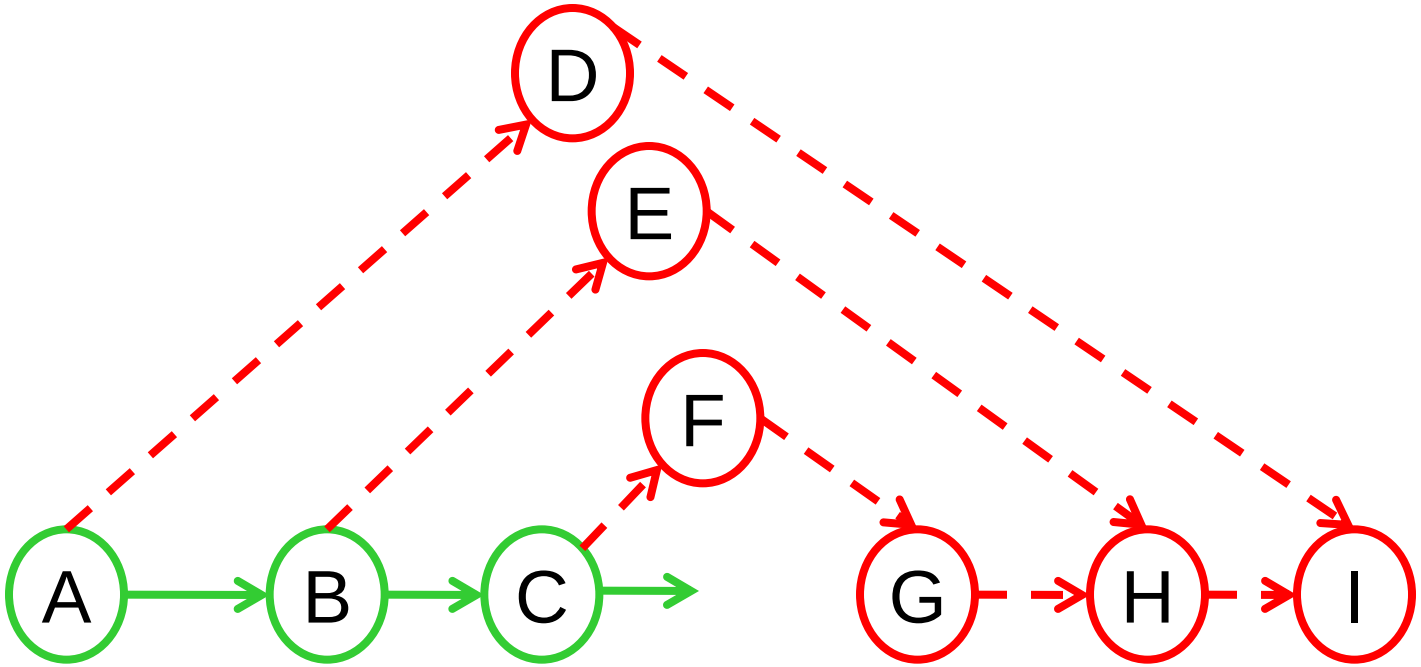
@IRQ (E, E)

linux//arch/x86/mm/pageattr.c:

static void **/* @IRQ (E, E) */ cpa_flush_array(…)**
{ … **BUG_ON(irqs_disabled()); …** }

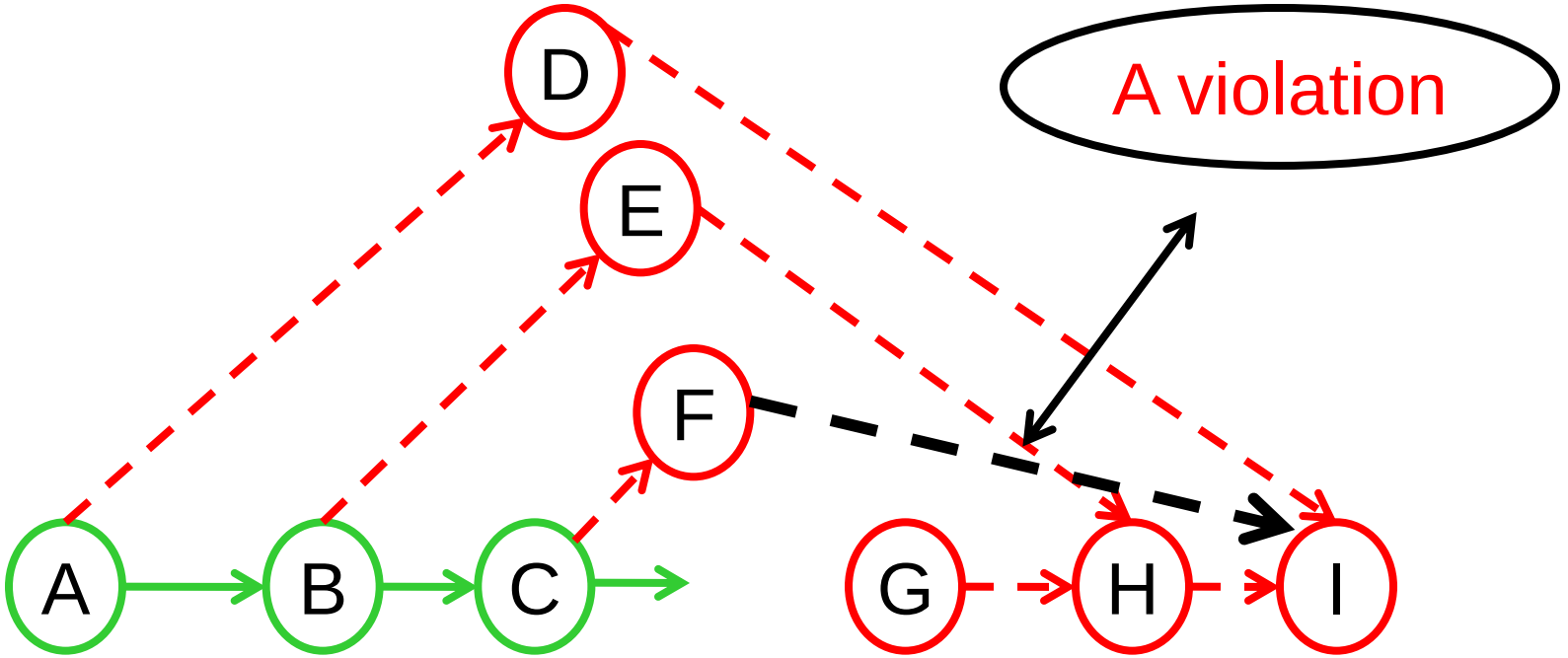
II . Related works

- Structure inferring
  - Hector[DSN 13]



II . Related works

- Structure inferring
  - Hector[DSN 13]



II . Related works

- Different to PF-Miner
  - Focus on kernel source code written by C program code
  - Paired functions between error handling paths and normal execution paths
  - Detects violations with extracted paired functions rather than programmers' notations or definitions

Agenda

I . Background

II. Related works

III. implementation for PF-Miner

IV. Experimental evaluation

V. Conclusion

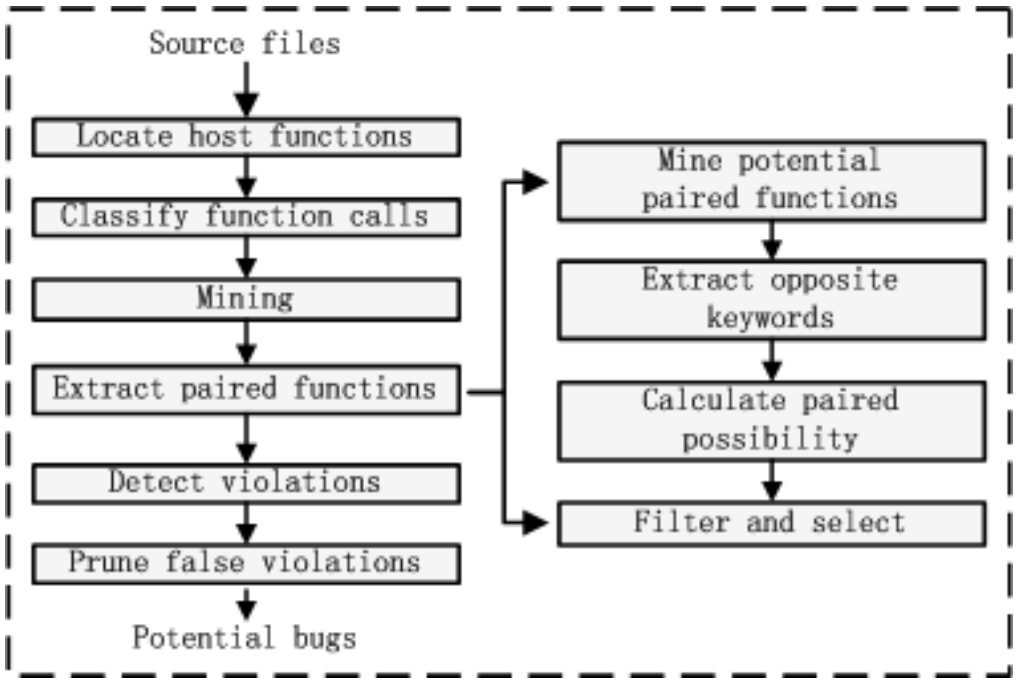
VI. References

III. implementation for PF-Miner

- Overview of PF-Miner
- Mining method in PF-Miner
- How to evaluate paired functions
- Extract paired functions
- Detect violations

III. implementation for PF-Miner

- Overview of PF-Miner
  - Take source code as input
  - Mining and extract paired functions
  - Detect violations



III. implementation for PF-Miner

- Mining method in PF-Miner
  - Host functions(H): they call paired functions
  - Called-function sets(C): called functions in H
  - Normal execution(N, n_i for i^{th} H)
  - Exception handling(E, E_i for i^{th} H)
  - Classification function(F_c):label function in error handling paths or normal paths

$$C_i \times F_c \rightarrow \{n_i, e_i\}^*$$

- Mining method in PF-Miner
  - **Decision function**(F_d):get potential paired functions

$$(n_i \times (n_e)^T) \times F_d \rightarrow [D_i]$$

$$D_i(a, b) > T_d \Rightarrow P_p \leftarrow P_p \cup \{(a, b)\}$$

  - **Selection function**(F_s):find out important paired functions with statistical methods

$$D_i(a, b) \times F_s > 0 \Rightarrow P_f \leftarrow P_f \cup \{(a, b)\}$$

III. implementation for PF-Miner

- Extract paired functions
  - Decision function
    - Two functions operate on same resources
    - Two functions execute opposite operations
    - Extract potential paired functions
  - Structure inferring
    - Assist the decision function to mine potential paired functions
  - Selection function
    - Reduce false positive
    - Associate statistical information
    - Filter from potential paired functions

III. implementation for PF-Miner

- Extract paired functions
  - Decision function
    - Functions with meaningful names and parameters
    - paired or single opposite keywords in their function names
    - pci_iomap / pci_iounmap with single opposite keyword “un”
    - pci_enable_device / pci_disable_device with paired opposite keyword
    - Except opposite keywords, rest substring is generally used to denote same resources

III. implementation for PF-Miner

- Extract paired functions
  - Structure inferring
    - Every error has its error handling path, and paired functions between them

Theory I:

$$\{\{a\},\{b\}\} \in \{n_i, e_i\}^*, \Rightarrow P_p \leftarrow P_p \cup \{(a,b)\}$$

Theory II:

$$\exists \{a^* \cup \{c\}, b^* \cup \{d\}\} \in \{n_i, e_i\}^*,$$
$$if\ \{a^*, b^*\} \in \{n_i, e_i\}^*, \Rightarrow\ P_p \leftarrow P_p \cup \{(c,d)\}$$

lemmas II-1:

$$\exists \{a^* \cup \{c^*\}, b^* \cup \{d^*\}\} \in \{n_i, e_i\}^*,$$
$$\{a^*, b^*\} \in \{n_i, e_i\}^* \Rightarrow \{n_i, e_i\}^* \leftarrow \{n_i, e_i\}^* \cup \{c^*, d^*\}$$

III. implementation for PF-Miner

- Extract paired functions
  - Structure inferring
  - An example

		Function name
1002	static int vortex_init_one(……)	
……	{	
1010	rc = pci_enable_device(pdev);	Normal execution
1011	if (rc < 0)	Error handling
1012	goto out;	Error handling
……	……	
1014	rc = pci_request_regions(pdev, DRV_NAME);	Normal execution
1015	if (rc < 0) {	Error handling
1016	pci_disable_device(pdev);	Error handling
1017	goto out;	Error handling
1018	}	Error handling
……	……	
1031	ioaddr = pci_iomap(pdev, pci_bar, 0);	Normal execution
……	……	
1034	if (!ioaddr) {	Error handling
1035	pci_release_regions(pdev);	Error handling
1036	pci_disable_device(pdev);	Error handling
1037	rc = -ENOMEM;	Error handling
1038	goto out;	Error handling
1039	}	

III. implementation for PF-Miner

- Detect violations
  - In an error handling path, the host function calls function a in its normal execution paths, while it does not call it paired function Pf(a) in later error handling paths
  - Pruning false positives
    - Search into the private interfaces in the host functions
    - Check paired function both in normal execution paths and error handling paths

Agenda

I . Background

II. Related works

III. implementation for PF-Miner

IV. Experimental evaluation

V. Conclusion

VI. References

IV. Experimental evaluation

- Computational complexity and time consumption
- Comparison between different versions
  - Capacity ability verification
- Detection results in version 3.10.10
  - Bug reports
- Detection on other large software
  - Extension ability to PF-Miner
- Discussion

IV. Experimental evaluation

- Computational complexity and time consumption
  - ABUS CM6331-C12C, Intel core i3-3220, 3.30 GHz, 4GB memory, 7200rpm SATA drive
  - N host functions, and calls M functions in average, about Thera$((n*m)^2)$
  - Cost about 150 seconds, located 8007 host functions for 4225 drivers

IV. Experimental evaluation

- Comparison between different versions
  - Ran 2.6.39 with PF-Miner, mine rules and detect violations
  - Ran 3.10.10 with same operations
  - Compare results between the two versions
    - 81 bugs are fixed when reported in 2.6.39
    - The commit messages in cgit are shown as
      - *disable pci device if there's an error after enabling it (fig.)*
      - *Release GPIO lines and IRQ on error in p54spi probe*
      - *Disable runtime PM in error paths from probe*

IV. Experimental evaluation

- Comparison between different versions

1988	static int cafe_pci_probe (……)	456	static int cafe_pci_probe (……)
……	{	……	{
233	cam = kzalloc(…);	467	cam = kzalloc(…);
	if (cam == NULL)	468	if (cam == NULL)
	goto out;	469	goto out;
……	……	……	……
2104	ret = pci_enable_device(pdev);	493	ret = pci_enable_device(pdev);
2105	if (ret)	494	if (ret)
2106	goto out_unreg;	495	goto out_free;
……	……	……	……
2043	cam->regs = pci_iomap(pdev, 0, 0);	499	mcam->regs = pci_iomap(pdev, 0, 0);
2044	if (! cam->regs) {	500	if (!mcam->regs) {
……	……	……	……
2046	goto out_unreg;	502	goto out_disable;
2047	}	503	}
……	……	……	……
2116	out_free:	535	out_disable:
2117	v4l2_device_unregister(&cam->v4l2_dev);	536	pci_disable_device(pdev);
2118	out_unreg:	537	out_free:
2119	kfree(cam);	538	kfree(cam);
2120	out:	539	out:
2121	return ret;	540	return ret;
2122	}	541	}
	(a) Bug occurred in kernel 2.6.39		(b) Bug fixed in kernel 3.10.0

IV. Experimental evaluation

- Detection results in version 3.10.10
  - Checks 4225 drivers
  - Extract 546 paired functions
  - Detect 983 violations
  - 15 bugs are confirmed by developers among the top 51 violations

Function names a	paired function b	$F_t(a,b)$	sum of violations
pci_enable_device	pci_disable_device	1056	63 violations
kzalloc	kfree	2556	430 violations
request_mem_region	release_mem_region	740	73 violations
pci_request_regions	pci_release_regions	562	1 violation

IV. Experimental evaluation

- Detection results in version 3.10.10
  - Two confirmed examples

```
1317  static int pm3fb_probe (……)
……    {
1325      err = pci_enable_device(dev);
……    ……
1351      if (!request_mem_region(…)) {
……            ……
1354          goto err_exit_neither;
1355      }
……    ……
1463  err_exit_neither:
1464      framebuffer_release(info);
1465      return retval;
1466  }
```

(a) Bug in *drivers/video/pm3fb.c*

```
64    static int stmmac_pci_probe (……)
……    {
89        addr = pci_iomap(pdev, i, 0);
……    ……
103       if (!priv) {
……            ……
106           goto err_out;
107       }
……    ……
117   err_out:
118       pci_clear_master(pdev);
……    ……
125   }
```

(b) Bug in *drivers/net/ethernet/stmicro/stmmac/stmmac_pci.c*

IV. Experimental evaluation

- Detection results in version 3.10.10
  - Automatically extract email address and their names from the source code
  - Merge emails when the developer has more than one related bugs to confirm
  - Send email with python script automatically
- Some developers say “well done, good job”, most email sends failed

IV. Experimental evaluation

- Detection on other large software
  - Modified less than 20 rows code in PF-Miner
  - Http server 2.4.7
    - Extracts 196 paired functions
    - apr_pool_create / apr_pool_destroy
    - Detect 11 violations
  - postgre SQL 9.3.2
    - Extracts 266 paired functions
    - Detects 12 violations

IV. Experimental evaluation

- Discussion
  - False positives
    - Call graph is too complex for static methods
    - Kernel interface functions are called indirectly
    - Called functions are related to sensitive variables
    - Some rules can not satisfy the paired rules completely, input_unregister_device contains input_free_device in fact.
  - Future works
    - Check type and parameters further
    - Tracing sensitive variables
    - Checking the whole call graph

Agenda

I . Background

II. Related works

III. implementation for PF-Miner

IV. Experimental evaluation

V. Conclusion

VI. References

V. Conclusion

- PF-Miner uses statistical method to efficiently and automatically extract paired functions between normal execution paths and error handling paths
- 81 bugs reported by PF-Miner in 2.6.39 have been fixed before the latest version 3.10.10
- PF-Miner only costs 150 seconds to extract rules and detect violations
- 983 violations are detected
- 15 bugs have been confirmed by developers so far among top 51 violations
- PF-Miner also can be used to detect other software with formal interfaces

Agenda

I . Background

II. Related works

III. implementation for PF-Miner

IV. Experimental evaluation

V. Conclusion

VI. References

VI. References(1)

[1] An empirical study of operating system errors. (SOSP 01)

[2] Tolerating Hardware Device Failures in Software.(SOSP 09)

[3] Dingo:Taming Device Driver.(Eurosys 09)

[4] Fault resilient drivers for Longhorn server.(Microsoft 04)

[5] Understanding modern device drivers.(ASPLOS 12)

[6] Unmodified device driver reuse and improved system dependability via virtual machines.(OSDI 04)

[7] The design and implementation of microdrivers.(SOSR 08)

[8] SafeDrive: Safe and recoverable extensions using language-based techniques.(OSDI 06)

[9] Fine-Grained Fault Tolerance using Device Checkpoints.(ASPLOS 13)

[10] SymDrive: testing drivers without devices.(OSDI 12)

[11] Software fault isolation with API integrity and multi-principal modules. (SOSP 11)

VI. References(2)

[12] Creating user-mode device drivers with a proxy.(NT Worshop 97)

[13] Improving the reliability of commodity operating systems(SOSR 03)

[14] S2E: A platform for in-vivo multi-path analysis of software systems.(ASPLOS 2011)

[15] Safe hardware access with the Xen virtual machine monitor.(OASIS 04)

[16] Fast byte-granularity software fault isolation.(SOSP 09)

[17] Windows XP kernel crash analysis.(LISA 06)

[18] Automatic device driver synthesis with Termite.(SOSP 09)

[19] Improving software diagnosability via log enhancement.(TOCS 12)

[20] PR-Miner: automatically extracting implicit programming rules and detecting violations in large software code(FSE 05)

[21] WYSIWIB: exploiting fine-grained program structure in a scriptable API-usage protocol-finding process. (SPE 13)

Acknowledgment

- Martin professor, Cardiff university
- Anonymous reviews
- The developers who help us confirm bugs and reviewed our bugs reports and patches
- PF-Miner is Supported by National High Technology Research and Development Program of China (Project Number 2011AA01A203)

Agenda

一．读博动机

二．研究内容介绍

一．Idea起源

二．研究框架

三．PF-Miner

三．研究感悟

四．实验演示

五．Q&A

三、研究感悟

- 天时、地利、人和
  - 动力、平台、自信、努力
    - 思想有多远你就能去多远
  - 树立远大的学术理想[学术大师]
    - 进入名校或研究所做研究
  - 进入行政服务部门[未来精英]
    - 基层挂职
    - 沟通协作职能培养
  - 积累产业界需要的知识[行业先锋]
    - 分布式系统
    - 数据挖掘·大数据分析·自然语言处理

Agenda

一．读博动机

二．研究内容介绍

- 一．Idea起源
- 二．研究框架
- 三．PF-Miner

三．研究感悟

四．实验演示

五．Q&A

四、实验演示

- PF-Miner实验过程简析
- PF-Miner针对3.10.10实验评估展示
- Python邮件脚本
  - 爱惜品牌
  - 严谨认真

感谢您的聆听

Thanks!

祝大家科研顺利，

早日毕业！