# openEuler内核之安全修复错误检测

慕冬亮

华中科技大学网络空间安全学院

# 个人介绍

- 慕冬亮，华中科技大学网安学院副教授
- 华中科技大学网安学院开放原子开源俱乐部责任人

- 研究对象：关键基础支撑软件（如操作系统内核）
- 研究内容
  - 基于漏洞生命周期的高效安全漏洞去除
  - 软件系统安全加固及其防御体系评估

- 邮件：dzm91@hust.edu.cn

# Google Open Source Peer Bonus Award

April 19, 2023

Dear Dongliang Mu,

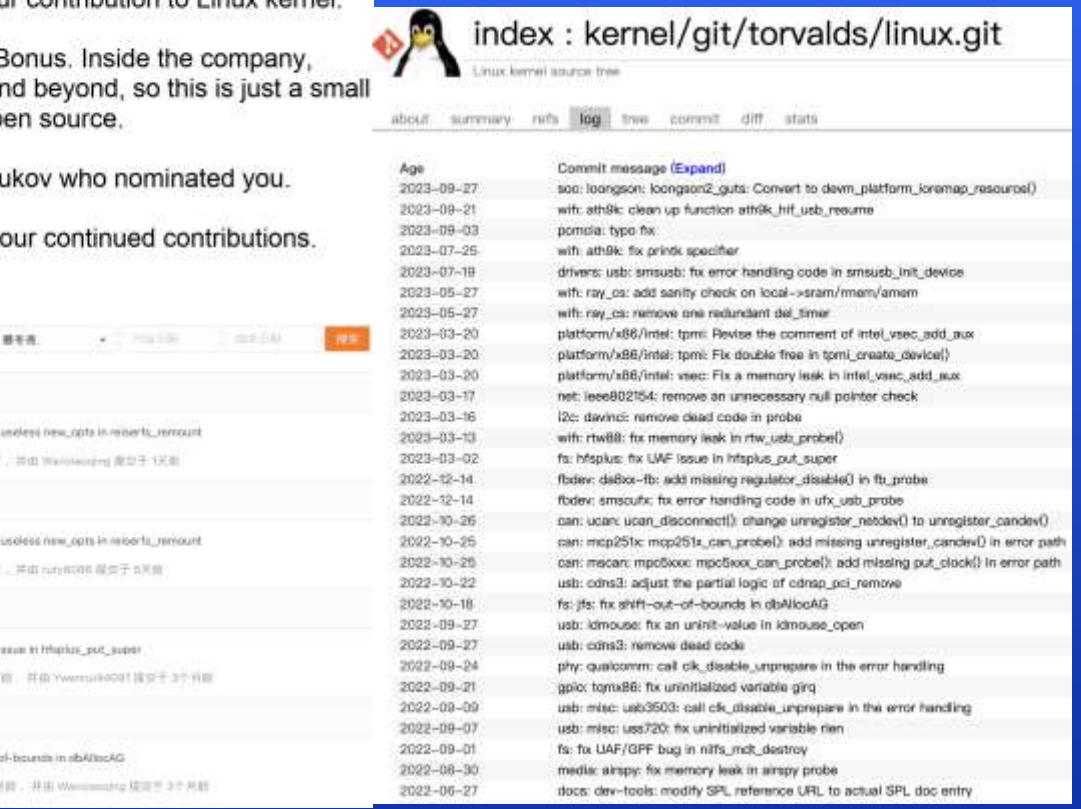
On behalf of Google Open Source, I would like to thank you for your contribution to Linux kernel.

We are honored to present you with a Google Open Source Peer Bonus. Inside the company, Googlers can give a similar bonus to each other for going above and beyond, so this is just a small way of saying thank you for your hard work and contributions to open source.

We hope you enjoy this gift from all of us at Google and Dmitry Vyukov who nominated you.

Thank you again for supporting open source! We look forward to your continued contributions.

Best regards,


[1] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/log/?qt=author&q=Dongliang+Mu

# 安全漏洞研究

漏洞发现 → 漏洞诊断 → 漏洞修复

围绕漏洞生命周期的三个阶段，高效去除关键基础软件中的安全漏洞

**硬件辅助二进制模糊测试**
- 硬件特性 (AsiaCCS'19)

**漏洞重现**
- 理解现实世界安全漏洞的可重现性 (USENIX SEC'18, TrustComm'21)

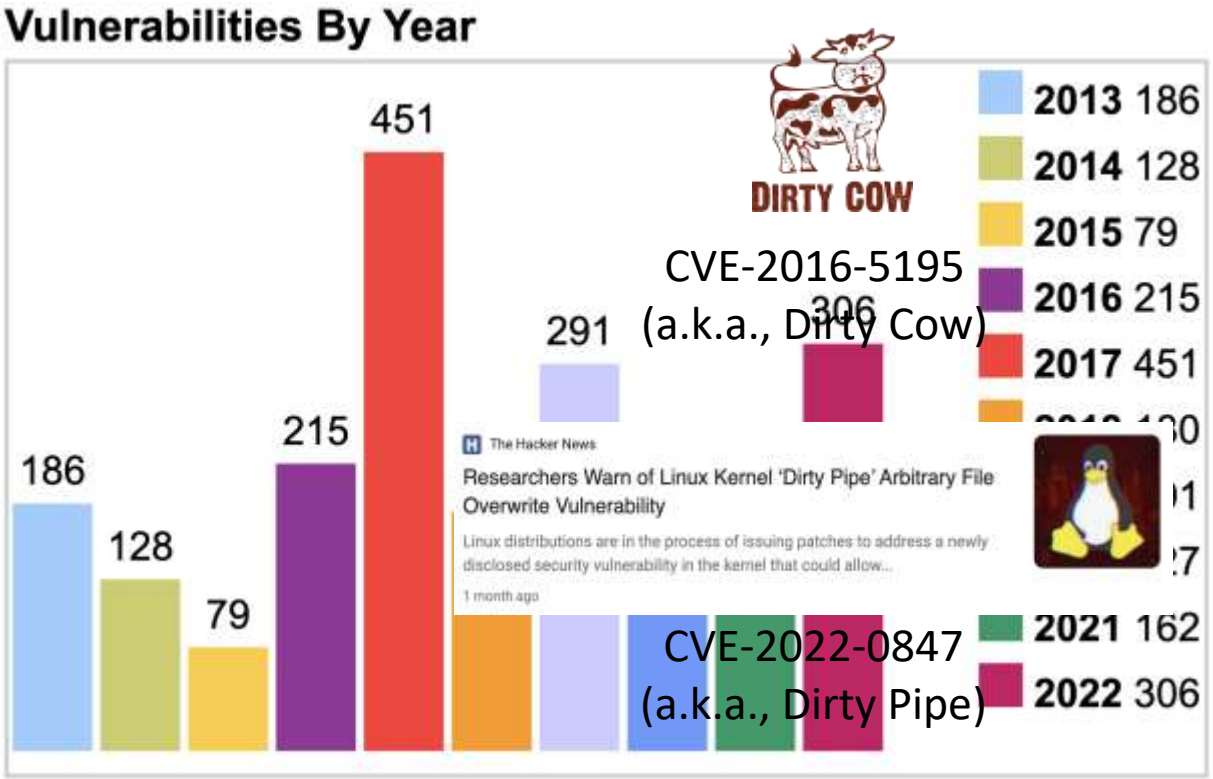
**自动化软件崩溃诊断**
- 程序分析 (ACM CCS'16, TSE'19)
- 硬件特性 (USENIX SEC'17)
- 深度学习 (USENIX SEC'19, ASE'19)

**软件崩溃去重**
- 内核崩溃报告分类与去重 (NDSS'22)

**软件漏洞危害评估**
- 探索新的错误行为来评估内核漏洞可利用性 (Oakland SP'22, TDSC'23)

**安全修复分析与测试**
- 内核安全修复的漏洞挖掘 (USENIX SEC'23)

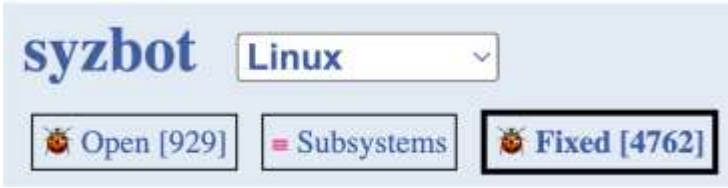
# 安全关键的Linux内核常包含漏洞


事实上，还有很多漏洞并未获得 CVE 编号

[1] https://e.huawei.com/cn/material/event/HC/37890f8a21ee46a9a18efd14f27db714

# Linux内核安全修复



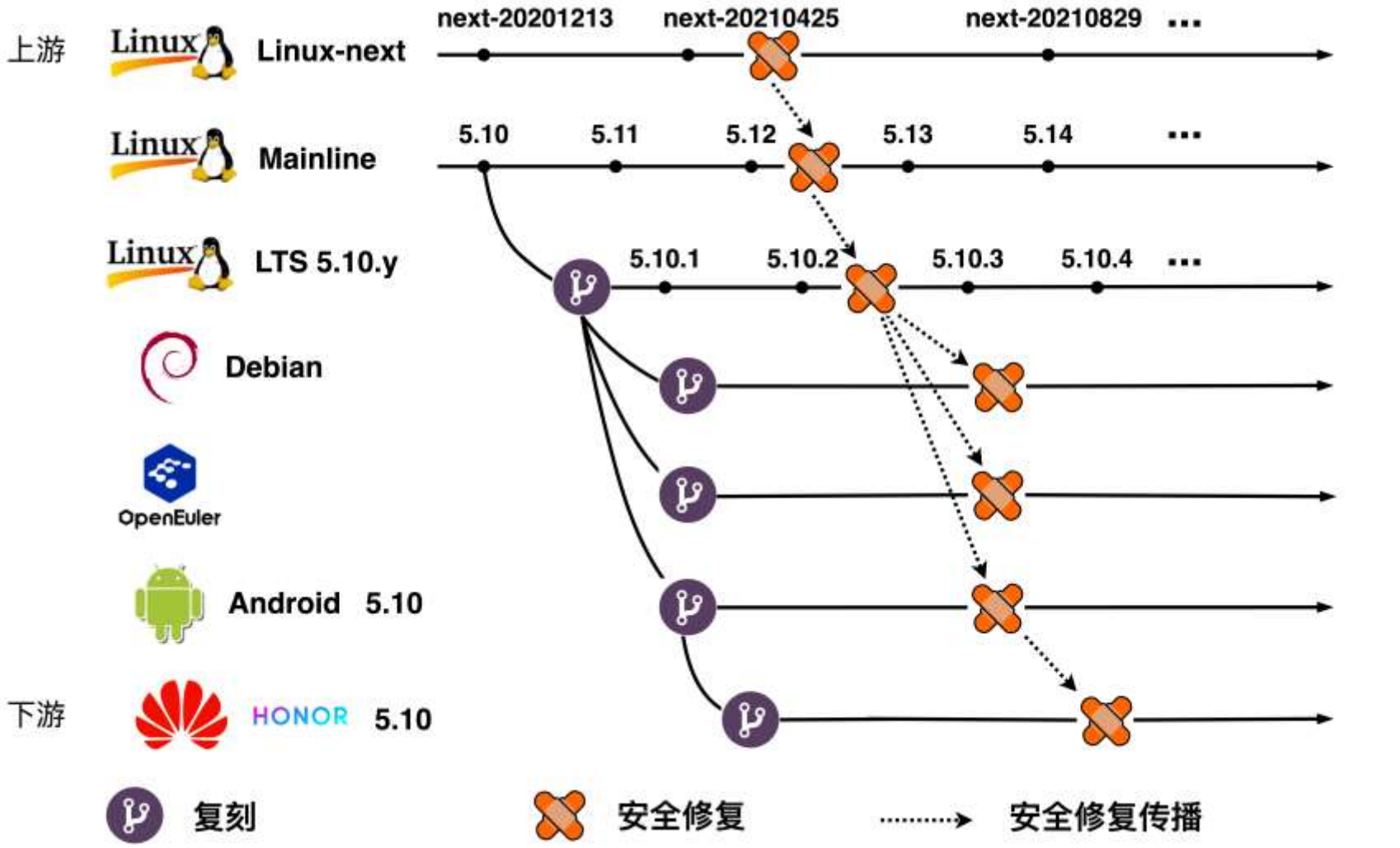


漏洞挖掘 漏洞上报 补丁编写 补丁测试 补丁合入

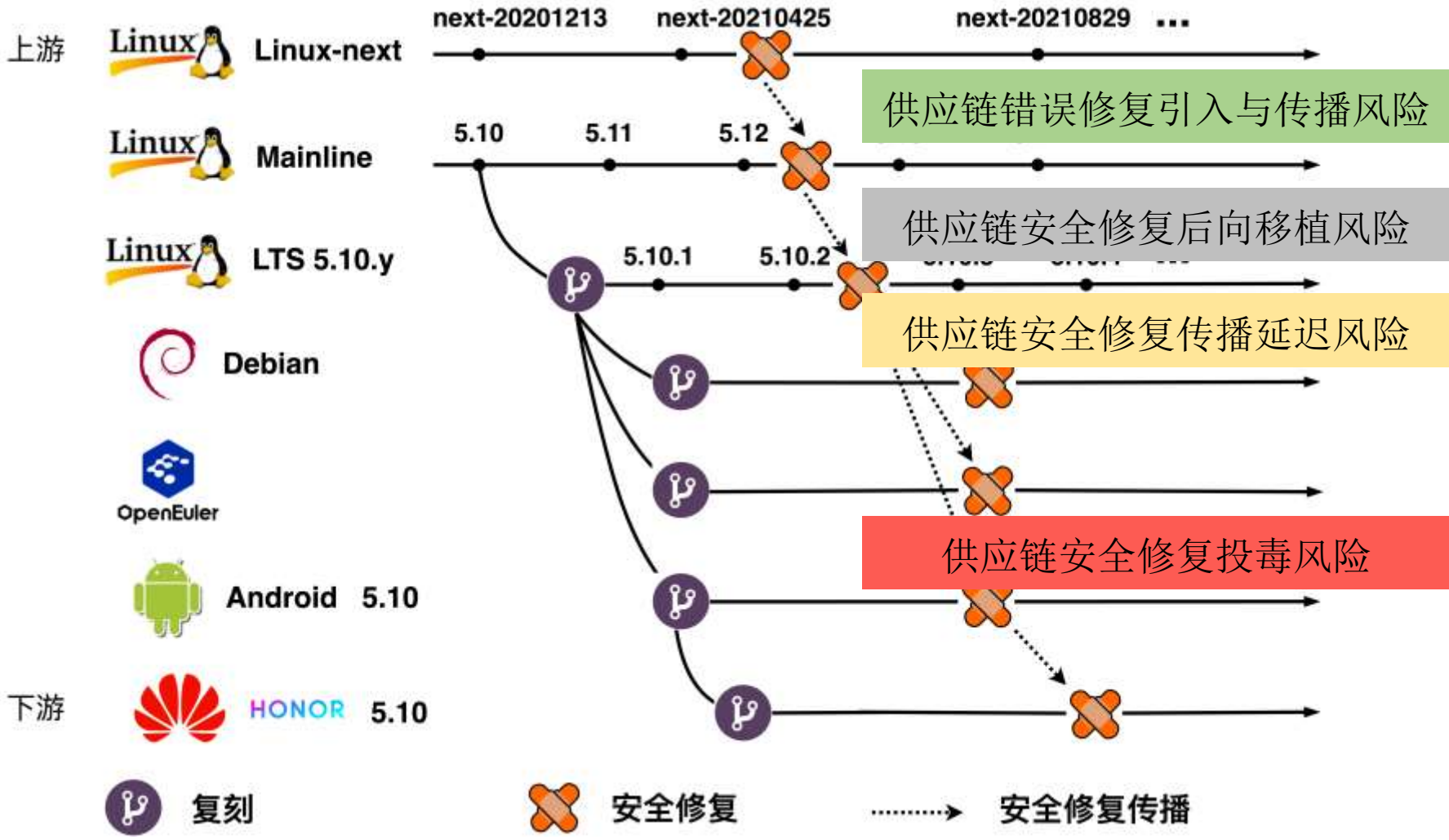
**Linux** 内核持续测试平台 **Syzbot** 使用的漏洞报告流程

| Title | Patch |
|---|---|
| KCSAN: data-race in drain_all_stock / drain_obj_stock (4) [cgroups] ... | 3b8abb323953 mm: kmem: fix a NULL pointer dereference in obj_stock_flush_required() |
| KASAN: use-after-free Read in qd_unlock (2) [gfs2] | f66af88e3321 gfs2: Stop using gfs2_make_fs_ro for withdraw |
| KASAN: slab-use-after-free Read in xsk_diag_dump [bpf] [net] | 3e019d8a05a3 xsk: Fix xsk_diag use-after-free error during socket cleanup |
| possible deadlock in static_key_slow_inc (3) [cgroups] | f0cc749254d1 cgroup,freezer: hold cpu_hotplug_lock before freezer_mutex in freezer_... |
| WARNING in try_grab_page [mm] | f443fd5af5db crypto, cifs: fix error handling in extract_iter_to_sg() |
| possible deadlock in raw_bind [can] | 11c9027c983e can: raw: fix lockdep issue in raw_release() |
| KASAN: slab-out-of-bounds Read in xlog_pack_data [xfs] | f1e1765aad7d xfs: journal geometry is not properly bounds checked |
| KCSAN: data-race in xfrmi_xmit / xfrmi_xmit (3) [net] | xfrm: interface: use DEV_STATS_INC() |
| upstream build error (20) [kernel] | 0a9567ac5e6a x86/mem_encrypt: Unbreak the AMD_MEM_ENCRYPT=n build |
| KASAN: use-after-free Read in j1939_session_get_by_addr | d966635b384b can: j1939: transport: make sure the aborted session will be deactivat... |
| BUG: sleeping function called from invalid context in alloc_buffer_h... | workingset: add missing rcu_read_unlock() in lru_gen_refault() |
| KMSAN: uninit-value in hwsim_cloned_frame_received_nl [wireless] | fba360a047d5 wifi: mac80211_hwsim: drop short frames |
| WARNING in ieee80211_probe_client [net] [wireless] | 67dfa589aa88 wifi: mac80211: check for station first in client probe |
| KMSAN: uninit-value in ieee80211_rx_handlers [wireless] [net] | 19e4a47ee747 wifi: mac80211: check S1G action frame size |
| WARNING in do_chunk_alloc [btrfs] | cd361199ff23 btrfs: wait on uncached block groups on every allocation loop |
| inconsistent lock state in btrfs_run_delayed_iputs [btrfs] | 866e98a4d95d btrfs: use irq_safe locking when running and adding delayed iputs |
| KASAN: slab-out-of-bounds Read in extract_iter_to_sg [crypto] | 4380499218c6 crypto: Fix af_alg_sendmsg(MSG_SPLICE_PAGES) sglist limit b6d972f68983... |

# 安全修复在Linux内核软件供应链上传播



# 内核软件供应链中漏洞修复安全风险



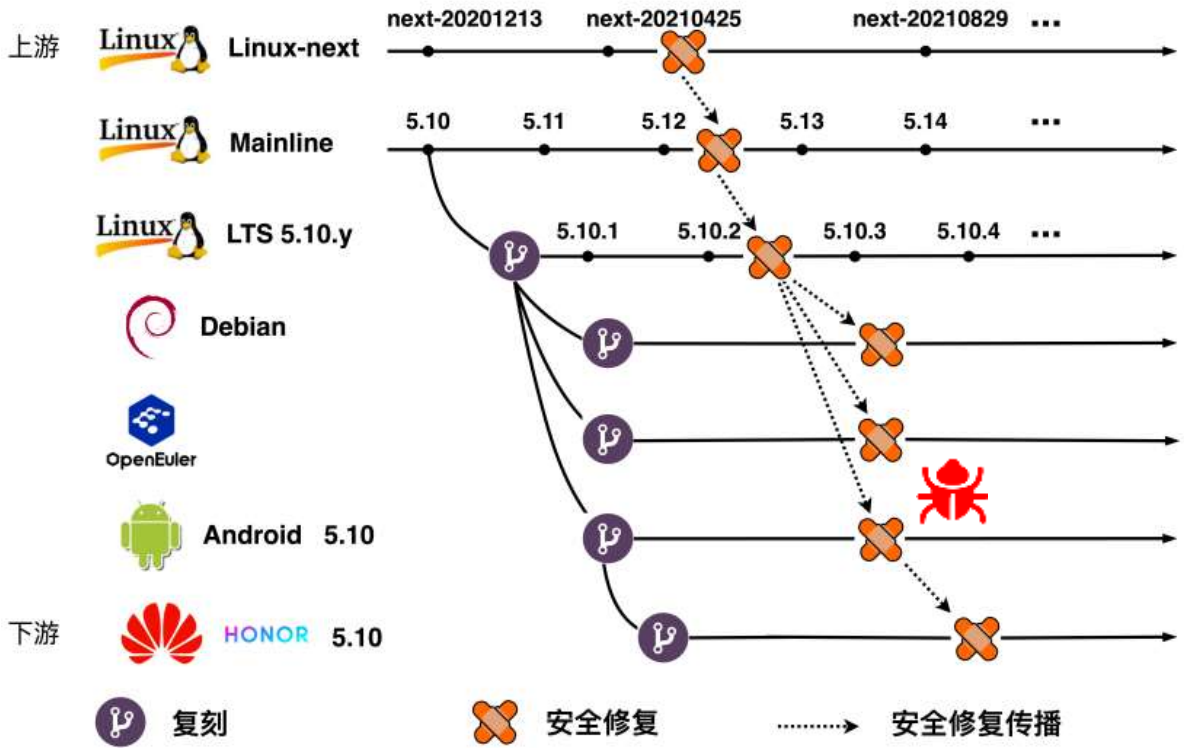
# 供应链错误安全修复引入与传播风险


Log4j itself was a case in point, with version 2.15.0 that fixed the original RCE vulnerability quickly found to have a different one. Hot on its heels came 2.16.0, which fixed the previous issue but had two other vulnerabilities. As of this writing, 2.17.1 is the recommended safe version – see our post on Log4Shell for a technical analysis of the original flaw. To give you one more example from recent months, in October 2021, a path traversal vulnerability was discovered in Apache web server 2.4.49. The fix rushed out in version 2.4.50 proved to be incomplete, and it took 2.4.51 to finally address the root cause.

[1] Has the Bug Really Been Fixed?, ICSE 10

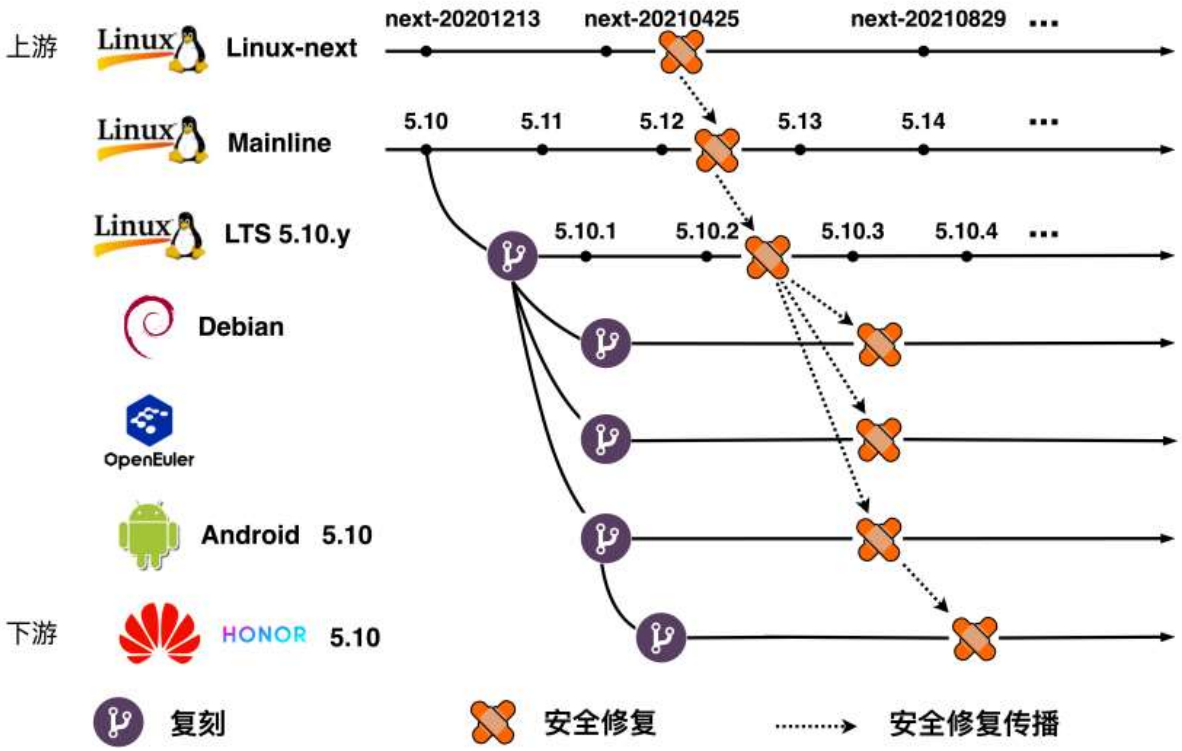
# 供应链安全修复后向移植风险

即使安全修复在引入时是正确的，它在供应链传播中也可能由于代码上下文等信息差异，成为错误安全修复



# 供应链安全修复传播延迟风险

几乎一半的 CVE 安全修复需要200天以上才能从上游主线传播到最终的 OEM 厂商，而10-30%需要1年以上时间进行修复[1]






[1] An Investigation of the Android Kernel Patch Ecosystem, USENIX SEC'21

# 供应链安全修复投毒风险

2021年明尼苏达大学因提交安全修复测试而被Linux内核社区屏蔽事件证明，攻击者完全可以提交错误安全修复进行供应链源头投毒，并影响下游国产操作系统。

## An open letter to the Linux community - April 24, 2021

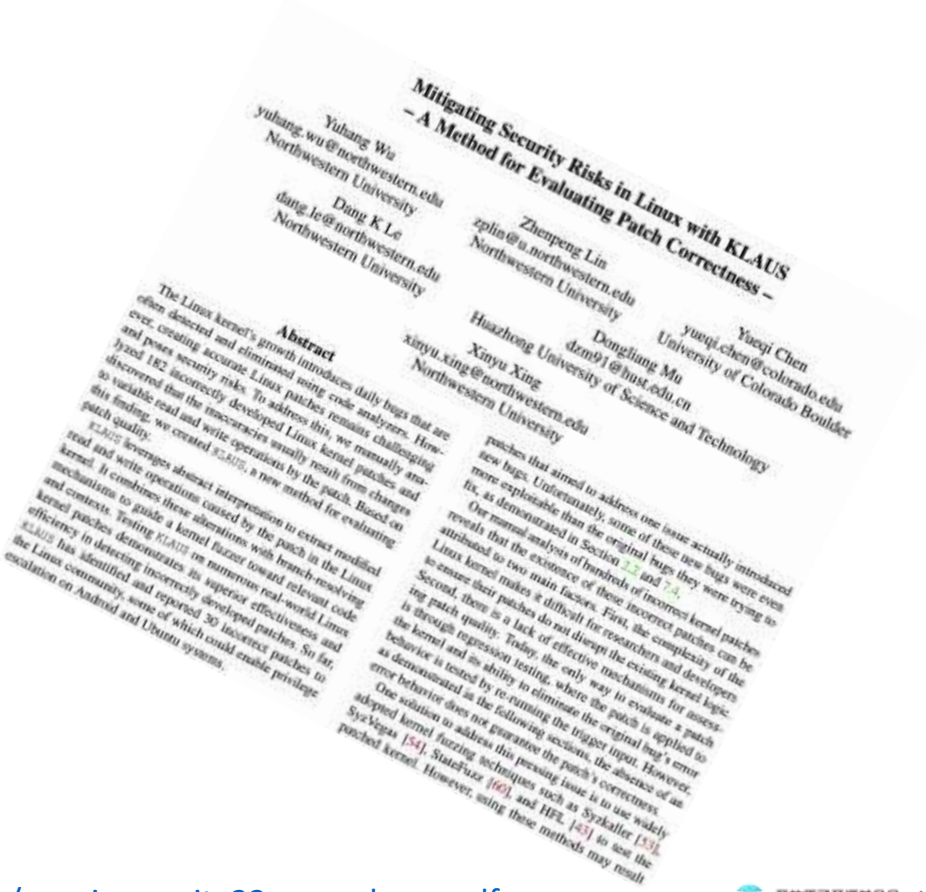
Dear Community Members:

We sincerely apologize for any harm our research group did to the Linux kernel community. Our goal was to identify issues with the patching process and ways to address them, and we are very sorry that the method used in the "hypocrite commits" paper was inappropriate. As many observers have pointed out to us, we made a mistake by not finding a way to consult with the community and obtain permission before running this study; we did that because we knew we could not ask the maintainers of Linux for permission, or they would be on the lookout for the hypocrite patches. While our goal was to improve the security of Linux, we now understand that it was hurtful to the community to make it a subject of our research, and to waste its effort reviewing these patches without its knowledge or permission.

We just want you to know that we would never intentionally hurt the Linux kernel community and never introduce security vulnerabilities. Our work was conducted with the best of intentions and is all about finding and fixing security vulnerabilities.

# 内核安全修复测试

Mitigating Security Risks in Linux with KLAUS : A Method for Evaluating Patch Correctness[1], USENIX SEC'23



Mitigating Security Risks in Linux with KLAUS
– A Method for Evaluating Patch Correctness –

Yuhang Wu
yuhang.wu@northwestern.edu
Northwestern University

Dang K Le
dang.le@northwestern.edu
Northwestern University

Zhenpeng Lin
zplin@u.northwestern.edu
Northwestern University

Yueqi Chen
yueqi.chen@colorado.edu
University of Colorado Boulder

Dongliang Mu
dzm91@hust.edu.cn
Huazhong University of Science and Technology

Xinyu Xing
xinyu.xing@northwestern.edu
Northwestern University

**Abstract**

The Linux kernel's growth introduces daily bugs that are often detected and eliminated using code analyzers. However, creating accurate Linux patches remains challenging and poses security risks. To address this, we manually analyzed 182 incorrectly developed Linux kernel patches and discovered that the inaccuracies usually result from changes to variable read and write operations by the patch. Based on this finding, we created KLAUS, a new method for evaluating patch quality.

KLAUS leverages abstract interpretation to extract modified read and write operations caused by the patch in the Linux kernel. It combines these alterations with branch-resolving mechanisms to guide a kernel fuzzer toward relevant code and contexts. Testing KLAUS on numerous real-world Linux kernel patches demonstrates its superior effectiveness and efficiency in detecting incorrectly developed patches. So far, KLAUS has identified and reported 30 incorrect patches to the Linux community, some of which could enable privilege escalation on Android and Ubuntu systems.

patches that aimed to address one issue actually introduced new bugs. Unfortunately, some of these new bugs were even more exploitable than the original bugs they were trying to fix, as demonstrated in Section 2.2 and 7.4.

Our manual analysis of hundreds of incorrect kernel patches reveals that the existence of these incorrect patches can be attributed to two main factors. First, the complexity of the Linux kernel makes it difficult for researchers and developers to ensure their patches do not disrupt the existing kernel logic. Second, there is a lack of effective mechanisms for assessing patch quality. Today, the only way to evaluate a patch is through regression testing, where the patch is applied to the kernel and its ability to eliminate the original bug's error behavior is tested by re-running the trigger input. However, as demonstrated in the following sections, the absence of an error behavior does not guarantee the patch's correctness.

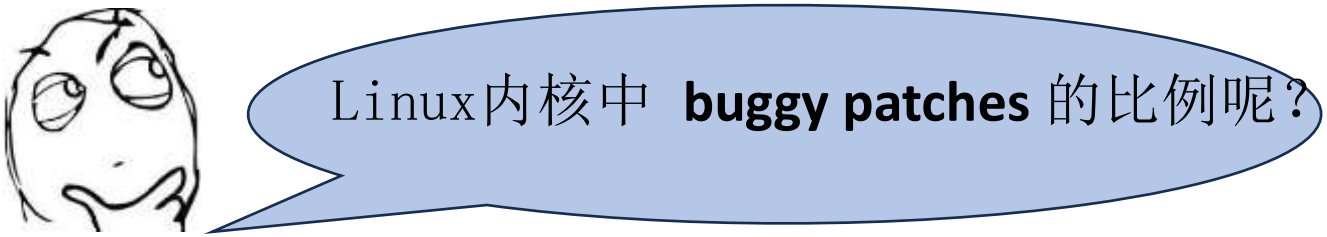
One solution to address this pressing issue is to use widely adopted kernel fuzzing techniques such as Syzkaller [53], SyzVegas [54], StateFuzz [46], and HFL [43] to test the patched kernel. However, using these methods may result



[1] https://www.usenix.org/system/files/usenixsecurity23-wu-yuhang.pdf

# Linux内核历史中buggy patch的比例

在 Ant, AspectJ, and Rhino三个项目中， buggy patches 占整体安全修复的 9% [1]



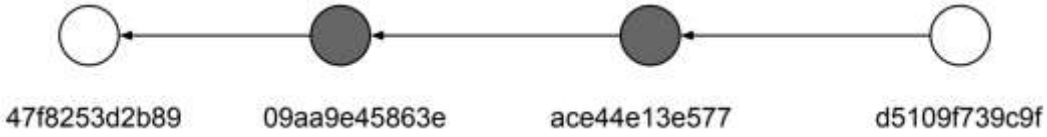

Author: Florian Westphal <fw@strlen.de>
Date: Sat Nov 17 11:32:29 2018 +0100

**netfilter: nfnetlink_cttimeout: fetch timeouts for udplite and gre, too**

syzbot was able to trigger the WARN in cttimeout_default_get() by passing UDPLITE as l4protocol. Alias UDPLITE to UDP, both use same timeout values ……

Fixes: 8866df9264a3 ("netfilter: nfnetlink_cttimeout: pass default timeout policy to obj_to_nlattr")
Signed-off-by: Florian Westphal <fw@strlen.de>

Methodology: 内核开发者利用“Fixes”标签来跟踪修复之前引入漏洞的代码提交，即，“Fixes”标签标识了引入内核漏洞的之前提交，而带有此标签的代码提交试图修复先前的内核漏洞。






[1] Has the Bug Really Been Fixed?, ICSE 10

# Linux内核历史中buggy patch的比例

- Fixes 标签自2011年开始，下列统计结果追踪到2022年

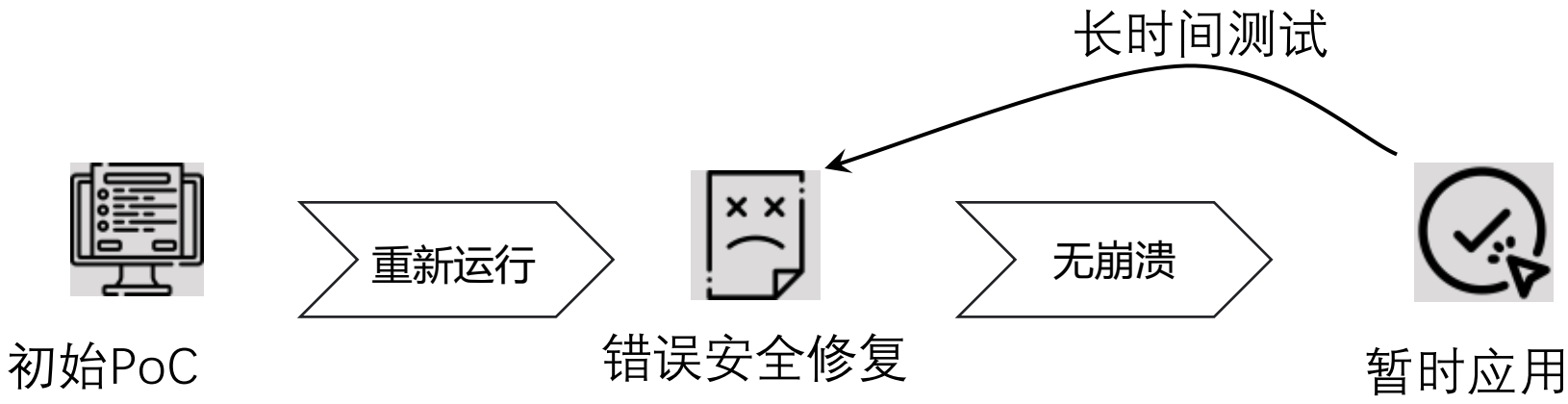
**Table 1:** Dataset Collection.

| Dataset | # of Collected Patch | # of Buggy Patch | Percentage |
|---|---|---|---|
| Fixes | 64023 | 3706 | 5.79% |
| CVE | 1420 | 124 | 8.73% |

CVE 判定 buggy patches 的方法：
1. 通过在 reference 中寻找是否存在两个以上的不同 commits
2. 通过在描述中寻找 incorrect 以及 incorrect 等关键
3. 利用Syzbot 中 Cause Bisection部分结果来串链

# Buggy patch 产生的原因

Buggy Patch 产生的原因：
1. 缺乏对漏洞相关代码的理解；
2. 漏洞根源的错误诊断；
普遍的安全修复错误：
1. 未考虑所有到达安全修复修改代码的路径；
2. 添加不足的合理性检查；



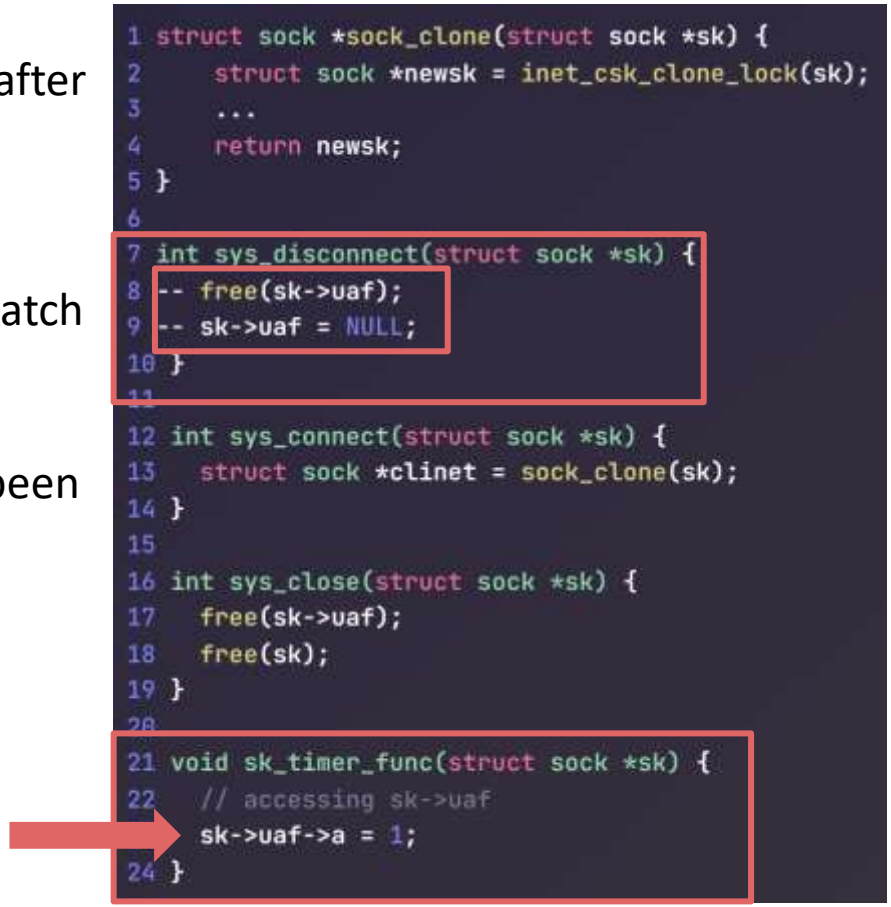
# 实际样例

**Initial UAF:** Dangling pointer in timer queue after sys_disconnect

**Incorrect patch:** line 8, 9 are deleted in the patch

**New UAF:** Dangling pointer left after sk has been cloned in sock_connect and after sys_close

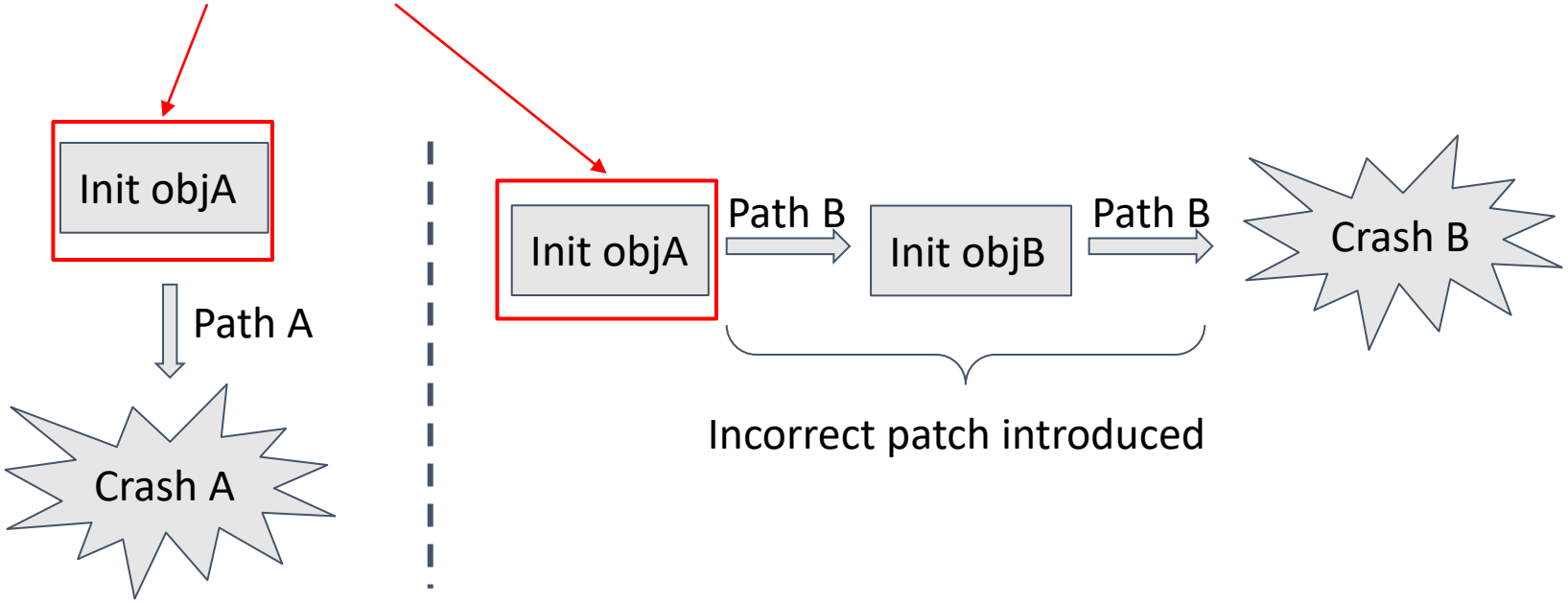
**Reason:** sk->uaf is not set to NULL



# AWRP (Altered Write-Read Pairs )

**我们手动分析 182 个内核错误修复，观察安全修复出错的原因**

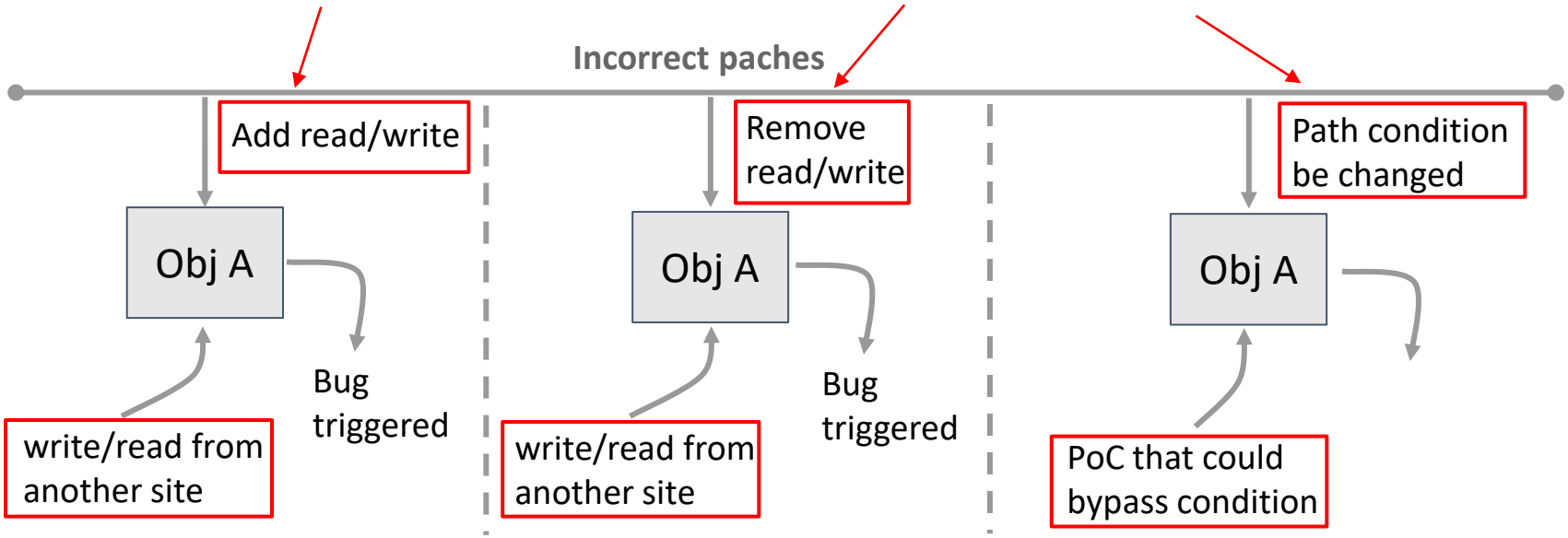
**观察一:** 旧漏洞和新漏洞之间共享相似的上下文



# AWRP (Altered Write-Read Pairs）

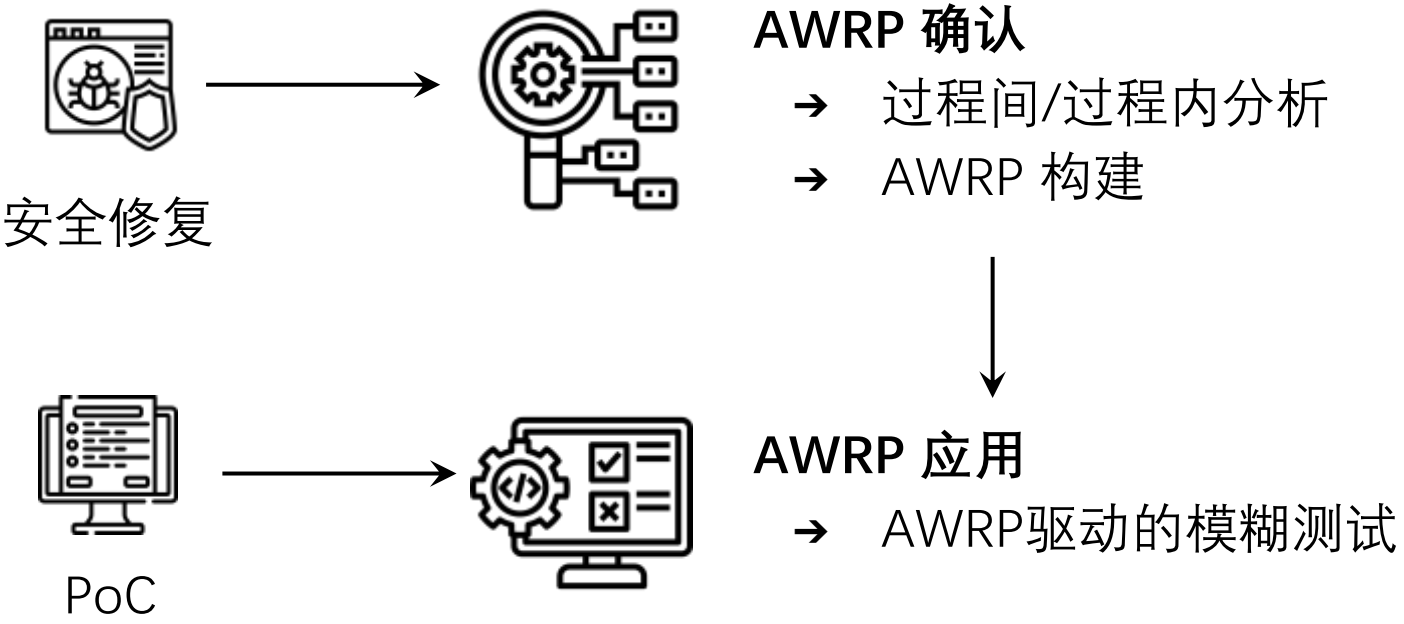
**我们手动分析 182 个内核错误修复，观察安全修复出错的原因**

**观察二:** 新漏洞由 Altered Write-Read Pairs (AWRP) 所导致



# KLAUS: 确定和应用AWRP的框架

AWRP 机制可提供安全修复的分析方法



# AWRP 驱动的内核模糊测试

- 基于Syzkaller开发
- 希望覆盖更多使用AWRP的地点
- 插装通向AWRP的主要路线上的基本块

两维覆盖率:
1. AWRP 覆盖率
2. 传统代码覆盖率

错误修复

# 实验评估

- 使用 Syzkaller 社区的 23 个真实样例
- 相同的初始种子
- 三天 & 5个轮次 & 相同环境
- 与 Syzkaller 进行对比

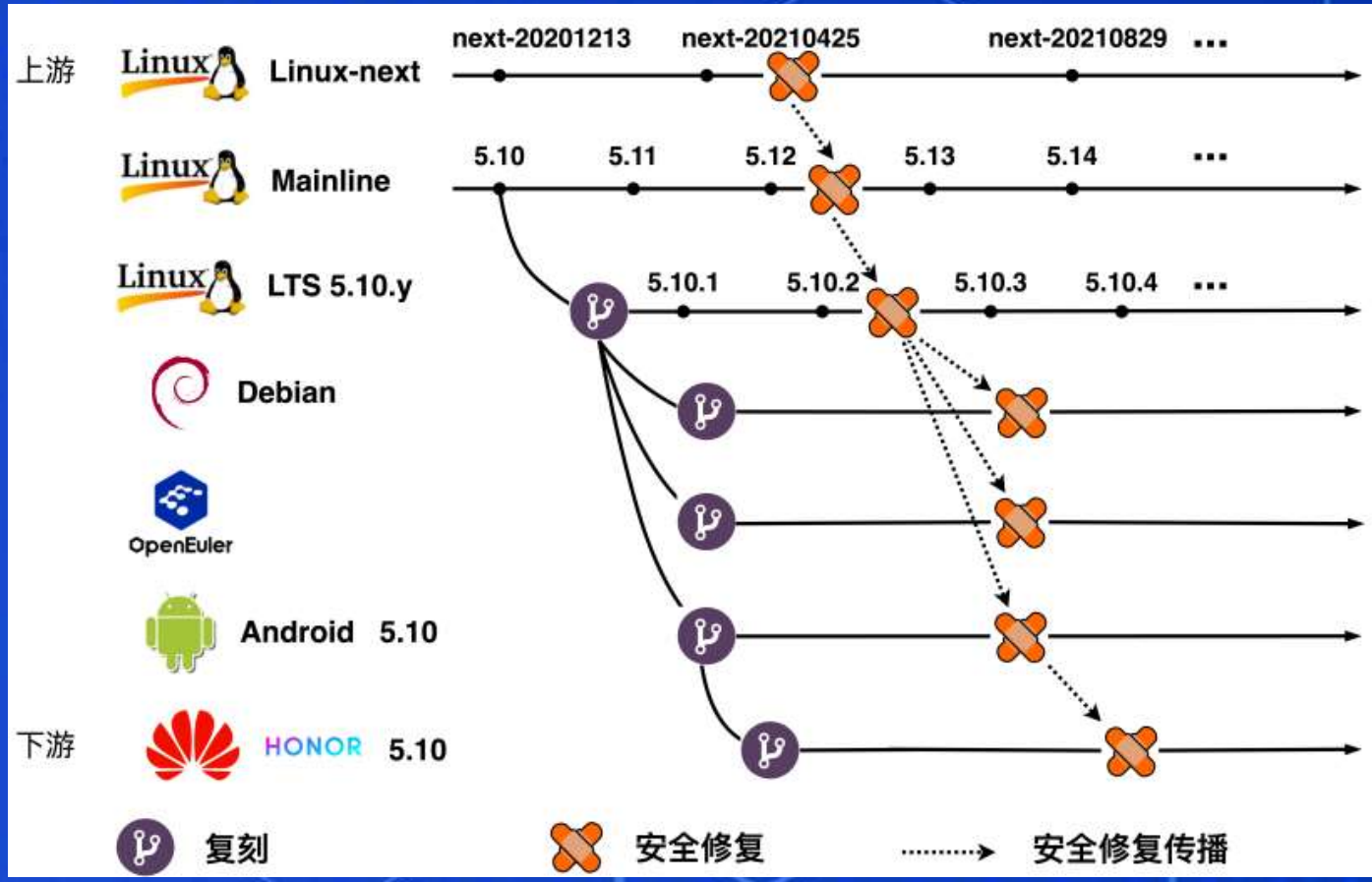
**KLAUS发现了 23/23 个错误安全修复**

**Syzkaller发现 13/23 个错误安全修复**

**KLAUS** triggers crashes caused by incorrect patches faster than **Syzkaller** in **12**/13 cases

**KLAUS** found **30** new incorrect patches in the wild! The community has confirmed and fixed **25** of these patches

# THANKS


上游
Linux
Linux-next
Linux
Mainline
Linux
LTS 5.10.y
Debian
OpenEuler
Android 5.10
下游
HONOR 5.10
next-20201213
next-20210425
next-20210829 ...
5.10
5.11
5.12
5.13
5.14 ...
5.10.1
5.10.2
5.10.3
5.10.4 ...
复刻
安全修复
安全修复传播