THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

Syzbot to Mainline

How I Merged 21 Kernel Patches as a First-Time Contributor

Deepanshu Kartikey
LFX Mentorship Graduate | Linux Kernel Contributor

About Me

- **Deepanshu Kartikey**
- Performance Engineer at ClickPost
  - Kernel-level observability: eBPF, perf, ftrace for production debugging
- LFX Mentorship Graduate (Fall 2025)
- Active Linux kernel contributor
- git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/log/?qt=author&q=deepanshu

The Problem

- **Kernel contribution feels impossible for newcomers**
- Millions of lines of code, decades of history
- Unfamiliar mailing-list workflow (no GitHub PRs)
- No clear starting point — where do you even begin?
- Most talks/resources assume deep kernel knowledge

The Answer: Syzbot

- **Syzbot = Google's 24/7 kernel fuzzer (powered by syzkaller)**
  - Generates random syscalls, throws them at the kernel, waits for it to crash
  - Found 1000s of real kernel bugs — open dashboard at syzkaller.appspot.com
- **How to break down any syzbot report (4 questions):**
  - 1. What kernel feature caught this? (KASAN, KMSAN, BUG_ON, lockdep)
  - 2. How is it configured? (CONFIG_KASAN=y, fuzzing options)
  - 3. What triggers the bug? (Read the reproducer step by step)
  - 4. Build the story: connect trigger → root cause → fix

The Workflow: 6 Steps

- **1. Pick a syzbot report**
  - Start with memory leaks, divide-by-zero, missing validations
- **2. Reproduce the bug**
  - Use the syzbot reproducer on a test VM
- **3. Read the code and understand the root cause**
  - git log, git blame, and the crash backtrace are your best friends
- **4. Write the fix**
- **5. Send the patch to the mailing list**
- **6. Iterate on reviewer feedback**

My Results: 3 Months, 21 Patches

- **Bug types fixed:**
  - Use-after-free (gfs2, mm/memfd)
  - Memory leaks (btrfs, gfs2)
  - Deadlocks (NFC networking)
  - Buffer overflows (tracing/DMA)
  - Information leaks (hugetlb)
  - Divide-by-zero (ocfs2, comedi)
  - Race conditions (ext4, hugetlbfs)
- **All found via syzbot. All merged to mainline.**

Real Example: ext4 Inline Data Bug

- **Syzbot reported: BUG_ON in ext4_write_inline_data()**
- **The crash:**
  - BUG_ON(pos + len > i_inline_size)
  - 0 + 4096 > 60 bytes → CRASH!
- **Reading the syzbot reproducer:**
  - Step 1: Mount ext4 filesystem with inline_data inode
  - Step 2: truncate(file, 50MB) — grows size, inline flag stays!
  - Step 3: sendfile() tries to write → triggers BUG_ON
- **Key insight: File has inline_data flag (60 byte capacity) but 50MB size!**

Root Cause & Finding the Fix

- **Call stack: sendfile() → ext4_write_inline_data() → CRASH**
  - Tries to write 4KB into 60 byte inline storage
- **Why? truncate() grew file to 50MB but didn’t clear inline flag**
- **How I found the fix location:**
  - grep -n ext4_setattr fs/ext4/*.c
  - VFS calls ext4_setattr() for all size changes
- **Added debug printk to confirm:**
  - setattr: old_size=10 new_size=50331645 has_inline=1
  - Inline flag still set after truncate — that’s the bug!

The Fix: 12 Lines of Code

- **In ext4_setattr(), before size change:**
  - Check: has_inline_data AND new_size > inline_capacity?
  - If yes: call ext4_convert_inline_data()
  - Converts inline storage → extent-based storage
- **Result: Inline data converted before size grows → No crash!**
- **Lessons from this bug:**
  - Read the crash stack trace carefully
  - Understand the reproducer step by step
  - Use printk to verify your hypothesis
  - Add validation at the right layer
- syzkaller.appspot.com/bug?extid=7de5fe447862fc37576f

Example 2: RDMA Invalid-Free Bug

- **KASAN reported: invalid-free in release_gid_table()**
  - Freeing 216 bytes inside a 224-byte object — the MIDDLE!
- **Reading the two stacks (alloc vs free):**
  - ALLOC: alloc_gid_table() → kzalloc_flex() → one single block
  - FREE: release_gid_table() → kfree(data_vec) + kfree(table) → two frees!
- **Key insight: Setup and teardown disagreed about memory layout**
- syzkaller.appspot.com/bug?extid=4334f9a250019c1b79b4

The Smoking Gun & Fix

- **kzalloc_flex() changed the memory layout:**
  - BEFORE: struct + data_vec = two separate allocations, two kfree()
  - AFTER: struct + data_vec = one flex array block, one kfree()
- **git log -S found the culprit commit:**
  - 74e2711bb2af “RDMA/core: Use kzalloc_flex for GID table”
  - Alloc was updated to flex, but free was forgotten!
- **The fix: delete 1 line**
  - Remove kfree(table->data_vec) — it’s inside the flex array
  - kfree(table) frees everything → Bug fixed!

Lessons from Code Review

- **Expect v2, v3, sometimes v4 revisions — that is normal**
- Maintainers care about correctness, not speed
- **Write detailed commit messages**
  - Explain what, why, and how. Include backtraces and reproduction steps
- **Be responsive and respectful**
  - Address every review comment, explain your reasoning
- **Reviewer feedback is the fastest way to level up**
  - I learned more from reviews than from any tutorial

How You Can Start Today

- **1. Visit syzkaller.appspot.com**
  - Filter by subsystem, look for open/unassigned bugs
- **2. Set up a kernel dev environment**
  - QEMU VM + latest kernel tree + syzbot reproducer
- **3. Start small**
  - Missing validations, NULL checks, error path leaks
- **4. Apply for LFX Mentorship**
  - Structured program with experienced mentors
- **5. Send your first patch!**
  - git format-patch, git send-email — the community is welcoming

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT INDIA

Thank You!

Deepanshu Kartikey | kartikey406@gmail.com
linkedin.com/in/deepanshu-kartikey-16024498
deepanshukartikey.dev