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 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

How to Start Your Kernel Journey

- **Three entry points — pick what suits you:**
- **1. Code style fixes**
  - Run checkpatch.pl on drivers/staging/
  - Fix formatting, indentation, coding style warnings
  - Easiest way in — no kernel knowledge needed
- **2. Bug fixes**
  - Pick a bug from syzkaller.appspot.com
  - Start with divide-by-zero, memory leaks, uninit values
  - You learn real debugging while fixing real bugs
- **3. Documentation**
  - Fix typos, improve explanations in Documentation/

What is Syzbot?

- **Syzbot = Google’s 24/7 kernel fuzzer (powered by syzkaller)**
  - Generates random syscalls, throws them at the kernel, waits for crash
  - Found 1000s of real kernel bugs
  - Open dashboard at syzkaller.appspot.com
- **Anyone can pick a bug and fix it**
- Today I’ll show you 5 real bugs I fixed — from easiest to hardest

Kernel Debugging Features

- **The kernel has built-in debugging features that catch bugs automatically**
- **Today we'll walk through each one, easiest to hardest:**
  - Oops report — kernel crashed, tells you where (divide-by-zero, NULL pointer)
  - kmemleak — finds memory allocated but never freed
  - BUG_ON — developer safety check, crashes kernel if impossible condition is true
  - KMSAN — catches reads of uninitialized memory
  - KASAN — catches bad memory access (use-after-free, invalid-free, out-of-bounds)
- **Each syzbot report tells you which feature caught the bug — right in the first line**

How to Read a Syzbot Report

- **6 steps to read any syzbot report:**
- **1. Check the subsystem**
  - Which part of the kernel? (ext4, networking, mm, etc.)
- **2. Check the bug type**
  - First line tells you: kmemleak, KMSAN, KASAN, BUG_ON, Oops?
- **3. Check RIP — the crash location**
  - Syzbot always shows the exact .c file and line number
- **4. Read the stack trace**
  - Bottom to top: what the user did → where the kernel crashed
- **5. Read the reproducer**
  - Find the 3-4 lines that matter, ignore the noise
- **6. Build the story**
  - Connect: trigger → root cause → fix

Level 1: Your Very First Kernel Patch

- **Everyone starts here — I did too (2023)**
- Watched Greg Kroah-Hartman’s video on writing your first patch
- Ran checkpatch.pl on drivers/staging/ — found coding style warnings
- **The fix:**
  - Moved } else if { to same line as closing brace
  - 7 lines added, 13 lines removed — pure code style cleanup
- **Even this took multiple patch revisions — and that’s okay!**
- Reviewed by Dan Carpenter, merged by Greg Kroah-Hartman
- **No tools. No debugging. Just read, fix, send (and resend)**

Level 1b: First Real Bug — Divide-by-Zero

- **Picked a syzbot bug — knew nothing about comedi drivers**
- **The bug:**
  - munge_chan %= chanlist_len — but chanlist_len was zero!
  - User sends command with chanlist_len=0 → kernel panics
- **The fix: add one check**
  - || async->cmd.chanlist_len == 0
  - 1 line changed. Divide-by-zero prevented. Kernel safe.
- Still took 2 patch revisions to get right
- **Lesson: You don’t need to understand the whole driver — just the bug**
- syzkaller.appspot.com/bug?extid=f6c3c066162d2c43a66c

Level 2: Memory Leak in ntfs3 (kmemleak)

- **New tool: kmemleak — finds memory allocated but never freed**
- Syzbot reported: BUG: memory leak in run_add_entry()
- **Reading the backtrace:**
  - link() → ntfs_link() → indx_insert_into_root() → indx_create_allocate()
  - attr_allocate_clusters() → run_add_entry() → kvmalloc() ← leaked!
- **The pattern:**
  - Alloc succeeds, something later fails, error path forgets to free
- Key question: who is supposed to free run memory?
- syzkaller.appspot.com/bug?extid=7adcddaeeb860e5d3f2f

Finding the Fix: Don’t Reinvent the Wheel

- **Success path: memcpy(&indx->alloc_run, &run) — ownership transfers**
- **Error path: run_deallocate() frees disk clusters — but who frees run memory?**
- **How I found the answer — grep the codebase:**
  - grep -n "alloc_run" fs/ntfs3/*.c
  - Found: run_close(&indx->alloc_run) used elsewhere for cleanup
- **The fix: add run_close(&run) at the out: label — 1 line added**
- **Lesson: grep existing code to find how the kernel already cleans up similar resources — follow the pattern**

Level 3: 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!**
- syzkaller.appspot.com/bug?extid=7de5fe447862fc37576f

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!**
- **I Googled “ext4 convert inline data” — found the function already existed**
- Grepped to see how it’s used, called it at the right place
- **Same pattern: don’t reinvent the wheel, find the existing function**

Level 4: KMSAN — Uninitialized Memory in iommufd

- **New tool: KMSAN (Kernel Memory Sanitizer)**
  - Catches reads of uninitialized memory
- Syzbot reported: uninit-value in batch_add_pfn_num()
- **KMSAN tells you exactly which field was read without being initialized**
  - The read: if (batch->kind != kind) — but batch->kind was never set!
- syzkaller.appspot.com/bug?extid=df28076a30d726933015

Tracing Backwards to the Fix

- **My first instinct: zero the entire struct — it worked!**
- But my mentor said: find the exact field that's missing
- **Traced the KMSAN stack backwards:**
  - KMSAN said: uninitialized read at batch->kind
  - Where does batch come from? → pfn_reader_init() → batch_init() → batch_clear()
  - Found it: batch_clear() sets end, pfns, npfns to 0 — but skips kind!
- **git blame confirmed: all from same commit (2022) — that's my Fixes tag**
- **The fix: add batch->kind = 0 — 1 line**
- **Lesson: don't zero everything — find the exact missing initialization**

Level 5: KASAN — RDMA Invalid-Free

- **New tool: KASAN — catches use-after-free, invalid-free, out-of-bounds**
- Syzbot reported: invalid-free in release_gid_table()
  - Freeing 216 bytes inside a 224-byte object — the MIDDLE!
- **KASAN gives you TWO stacks — alloc and 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 Fix & Open Source Reality

- **git log -S "kzalloc_flex" found the culprit commit:**
  - Alloc was refactored from two blocks to one — but free was never updated!
- **The fix: delete kfree(table->data_vec) — 1 line**
- **Plot twist: someone else submitted the same fix before me**
- But my solution was identical — proof the debugging was correct
- **Lesson: In open source, speed matters. Even if you lose the race, you proved your skill. Keep going.**

Writing a Good Commit Message

- **Subject line: subsystem: short description**
  - e.g., ext4: convert inline data to extents when truncate exceeds inline size
- **Paragraph 1: Describe the bug — what goes wrong and how**
- **Paragraph 2: Describe the fix — what you added or changed**
- **Paragraph 3 (optional): Why this fix is correct**
- **Essential tags:**
  - Reported-by: syzbot+xxx@syzkaller... — who reported it
  - Closes: syzbot link — links to the bug report
  - Fixes: commit hash ("title") — which commit introduced the bug
  - Cc: stable@kernel.org — gets your fix backported to older kernels
  - Signed-off-by: Your Name — your sign-off

How to Send Your Patch

- **Find the right maintainer:**
  - ./scripts/get_maintainer.pl your_patch.patch
  - Shows who to CC — maintainer + mailing list
- **Check your patch before sending:**
  - ./scripts/checkpatch.pl --strict your_patch.patch
  - Catches style issues, missing tags, formatting errors
- **Send it:**
  - git format-patch -1 → creates the patch file
  - git send-email your_patch.patch → sends to mailing list
- **Always fix checkpatch warnings before sending!**

How You Can Start Today

- **1. Pick a bug from syzkaller.appspot.com**
  - Start with the easiest: KMSAN uninit-value, divide-by-zero, memory leaks
  - Don’t worry about understanding the whole subsystem
- **2. Fix it**
  - Read the stack trace, grep the codebase, follow existing patterns
  - You’ve seen 5 examples today — you have the tools
- **3. Run checkpatch.pl and find the maintainer**
- **4. Send it and wait for review**
  - v1 might get rejected — that’s normal (mine did too!)
  - Learn from feedback, send v2 — that’s how you grow
- **5. Apply for LFX Mentorship**

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT INDIA

Thank You!

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