Reproducing a syzbot Bug in 5 Minutes — Now with virtme-ng!

Roman "Hedin" Storozhenko <romeusmeister@gmail.com>

Feb.1, 2026

FOSDEM'26

Who am I

About

- Ocassional Linux Kernel contributor [patches]
- Speaker at Open Source Summit Europe 2025. Amsterdam, Netherlands.
- Technical author. Hackathon mentor & judge (e.g., Globee Awards).
- SW engineer at Intel®, focused on enabling Intel® Xeon® RDT features
- Regular user of virtme-ng

Contacts

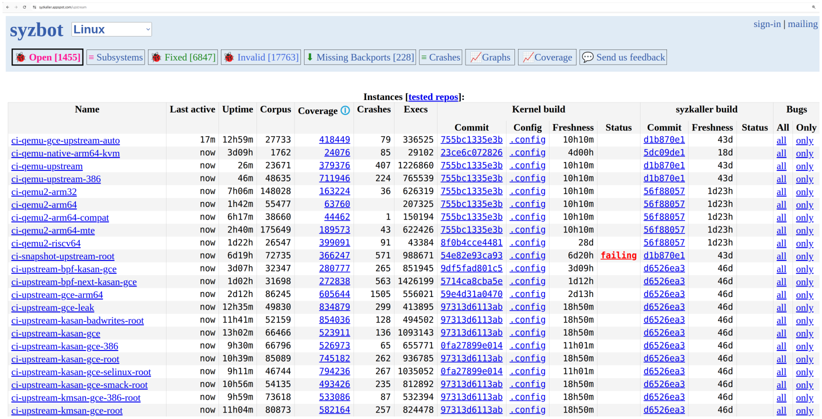
LinkedIn: https://www.linkedin.com/in/roman-st/
Email: romeusmeister@gmail.com

Agenda

1. Introduction to syzbot and kernel fuzzing
2. The bug we're going to reproduce (kmemleak)
3. virtme-ng: Why it accelerates kernel development
4. Clarification: What "5 minutes" really means
5. Live demo — reproducing & fixing a real syzbot bug
6. When virtme-ng shines (and when it doesn't)
7. Questions & discussion

Bug Source: syzbot

- Syzkaller is an unsupervised coverage-guided kernel fuzzer
- Syzbot - CI continuously fuzzes main Linux kernel branches and automatically reports found bugs to kernel mailing lists

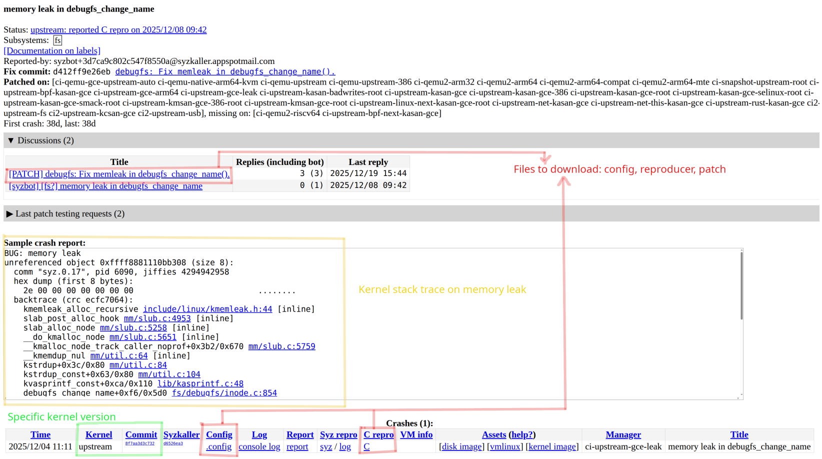


Instances [tested repos]:

Name	Last active	Uptime	Corpus	Coverage	Crashes	Execs	Kernel build				syzkaller build			Bugs	
							Commit	Config	Freshness	Status	Commit	Freshness	Status	All	Only
ci-qemu-gce-upstream-auto	17m	12h59m	27733	418449	79	336525	755bc1335e3b	.config	10h10m		d1b870e1	43d		all	only
ci-qemu-native-arm64-kvm	now	3d09h	1762	24076	85	29102	23ce6c072826	.config	4d00h		5dc09de1	18d		all	only
ci-qemu-upstream	now	26m	23671	379376	407	1226860	755bc1335e3b	.config	10h10m		d1b870e1	43d		all	only
ci-qemu-upstream-386	now	46m	48635	711946	224	765539	755bc1335e3b	.config	10h10m		d1b870e1	43d		all	only
ci-qemu2-arm32	now	7h06m	148028	163224	36	626319	755bc1335e3b	.config	10h10m		56f88057	1d23h		all	only
ci-qemu2-arm64	now	1h42m	55477	63760		207325	755bc1335e3b	.config	10h10m		56f88057	1d23h		all	only
ci-qemu2-arm64-compat	now	6h17m	38660	44462	1	150194	755bc1335e3b	.config	10h10m		56f88057	1d23h		all	only
ci-qemu2-arm64-mte	now	2h40m	175649	189573	43	622426	755bc1335e3b	.config	10h10m		56f88057	1d23h		all	only
ci-qemu2-riscv64	now	1d22h	26547	399091	91	43384	8f0b4cce4481	.config	28d		56f88057	1d23h		all	only
ci-snapshot-upstream-root	now	6d19h	72735	366247	571	988671	54e82e93ca93	.config	6d20h	failing	d1b870e1	43d		all	only
ci-upstream-bpf-kasan-gce	now	3d07h	32347	280777	265	851945	9df5fad801c5	.config	3d09h		d6526ea3	46d		all	only
ci-upstream-bpf-next-kasan-gce	now	1d02h	31698	272838	563	1426199	5714ca8cba5e	.config	1d12h		d6526ea3	46d		all	only
ci-upstream-gce-arm64	now	2d12h	86245	605644	1505	556021	59e4d31a0470	.config	2d13h		d6526ea3	46d		all	only
ci-upstream-gce-leak	now	12h35m	49830	834879	299	413895	97313d6113ab	.config	18h50m		d6526ea3	46d		all	only
ci-upstream-kasan-badwrites-root	now	11h41m	52159	854036	128	494502	97313d6113ab	.config	18h50m		d6526ea3	46d		all	only
ci-upstream-kasan-gce	now	13h02m	66466	523911	136	1093143	97313d6113ab	.config	18h50m		d6526ea3	46d		all	only
ci-upstream-kasan-gce-386	now	9h30m	66796	526973	65	655771	0fa27899e014	.config	11h01m		d6526ea3	46d		all	only
ci-upstream-kasan-gce-root	now	10h39m	85089	745182	262	936785	97313d6113ab	.config	18h50m		d6526ea3	46d		all	only
ci-upstream-kasan-gce-selinux-root	now	9h11m	46744	794236	267	1035052	0fa27899e014	.config	11h01m		d6526ea3	46d		all	only
ci-upstream-kasan-gce-smack-root	now	10h56m	54135	493426	235	812892	97313d6113ab	.config	18h50m		d6526ea3	46d		all	only
ci-upstream-kmsan-gce-386-root	now	9h59m	73618	533086	87	532394	97313d6113ab	.config	18h50m		d6526ea3	46d		all	only
ci-upstream-kmsan-gce-root	now	11h04m	80873	582164	257	824478	97313d6113ab	.config	18h50m		d6526ea3	46d		all	only

The Bug We'll Reproduce

This bug was caught by a reproducer using the kmemleak kernel module (kernel's valgrind)


Files to download: config, reproducer, patch
Kernel stack trace on memory leak
Specific kernel version

memory leak in debugfs_change_name

Status: upstream: reported C repro on 2025/12/08 09:42
Subsystems: fs
[Documentation on labels]
Reported-by: syzbot+3d7ca9c802c547f8550a@syzkaller.appspotmail.com
Fix commit: `d412ff9e26eb` `debugfs: Fix memleak in debugfs_change_name().`
Patched on: [ci-qemu-gce-upstream-auto ci-qemu-native-arm64-kvm ci-qemu-upstream ci-qemu-upstream-386 ci-qemu2-arm32 ci-qemu2-arm64 ci-qemu2-arm64-compat ci-qemu2-arm64-mte ci-snapshot-upstream-root ci-upstream-bpf-kasan-gce ci-upstream-gce-arm64 ci-upstream-gce-leak ci-upstream-kasan-badwrites-root ci-upstream-kasan-gce ci-upstream-kasan-gce-386 ci-upstream-kasan-gce-root ci-upstream-kasan-gce-selinux-root ci-upstream-kasan-gce-smack-root ci-upstream-kmsan-gce-386-root ci-upstream-kmsan-gce-root ci-upstream-linux-next-kasan-gce-root ci-upstream-net-kasan-gce ci-upstream-net-this-kasan-gce ci-upstream-rust-kasan-gce ci2-upstream-fs ci2-upstream-kcsan-gce ci2-upstream-usb], missing on: [ci-qemu2-riscv64 ci-upstream-bpf-next-kasan-gce]
First crash: 38d, last: 38d

▼ Discussions (2)

Title	Replies (including bot)	Last reply
[PATCH] debugfs: Fix memleak in debugfs_change_name().	3 (3)	2025/12/19 15:44
[syzbot] [fs?] memory leak in debugfs_change_name	0 (1)	2025/12/08 09:42

▶ Last patch testing requests (2)

Sample crash report:
```
BUG: memory leak
unreferenced object 0xffff8881110bb308 (size 8):
  comm "syz.0.17", pid 6090, jiffies 4294942958
  hex dump (first 8 bytes):
    2e 00 00 00 00 00 00 00                          ........
  backtrace (crc ecfc7064):
    kmemleak_alloc_recursive include/linux/kmemleak.h:44 [inline]
    slab_post_alloc_hook mm/slub.c:4953 [inline]
    slab_alloc_node mm/slub.c:5258 [inline]
    __do_kmalloc_node mm/slub.c:5651 [inline]
    __kmalloc_node_track_caller_noprof+0x3b2/0x670 mm/slub.c:5759
    __kmemdup_nul mm/util.c:64 [inline]
    kstrdup+0x3c/0x80 mm/util.c:84
    kstrdup_const+0x63/0x80 mm/util.c:104
    kvasprintf_const+0xca/0x110 lib/kasprintf.c:48
    debugfs_change_name+0xf6/0x5d0 fs/debugfs/inode.c:854
```

Crashes (1):

Time	Kernel	Commit	Syzkaller	Config	Log	Report	Syz repro	C repro	VM info	Assets (help?)	Manager	Title
2025/12/04 11:11	upstream	8f7aa3d3c732	d6526ea3	.config	console log	report	syz / log	C		[disk image] [vmlinux] [kernel image]	ci-upstream-gce-leak	memory leak in debugfs_change_name

virtme-ng: Make Linux Kernel Development Fun Again

Why?

virtme-ng significantly decreases VM load time and allows you to forget about userspace and focus on kernel development

virtme-ng "formula"

virtme-ng = QEMU/KVM lightweight VM + Linux Kernel + CoW host filesystem mounted via virtiofs + minimal userspace init

Development cycles comparison

Traditional approach

- Rebuild Kernel
- Rebuild VM image containing OS with new kernel (or load new kernel into VM userspace)
- Boot VM, initialize userspace, and log in
- Test
- Fix
- Repeat

virtme-ng

- Rebuild Kernel
- Run virtme-ng with new Kernel
- Test
- Fix
- Repeat

What ‘5 minutes’ really means

Important Clarification

- A Linux kernel does not compile from scratch in 5 minutes
- virtme-ng does not speed up compilation
- The speedup is in the test iteration, not the build

What is fast

- Rebuilding after a small patch
- Booting and testing immediately
- No userspace reinstall or VM image rebuild
- No userspace initialization and login

virtme-ng optimizes the build → run → test loop, not the initial build.

Demo setup dir structure: kernel source tree

Kernel source tree checked out at the commit containing the bug:

```
hedin@laptop:~/prj/linux$ git log -1

commit 8f7aa3d3c7323f4ca2768a9e74ebbe359c4f8f88 (HEAD)
Merge: 015e7b0b0e8e 4de44542991e
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Wed Dec 3 17:24:33 2025 -0800

    Merge tag 'net-next-6.19' of
         git://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next

    Pull networking updates from Jakub Kicinski:
     "Core & protocols:

       - Replace busylock at the Tx queuing layer with a lockless list.

         Resulting in a 300% (4x) improvement on heavy TX workloads, sending
         twice the number of packets per second, for half the cpu cycles.

       - Allow constantly busy flows to migrate to a more suitable CPU/NIC
         queue.

         Normally we perform queue re-selection when flow comes out of idle,
         but under extreme circumstances the flows may be constantly busy.

         Add sysctl to allow periodic rehashing even if it'd risk packet
         reordering.
        ......................................................................
```

Demo setup dir structure: ccache, kernel build, reproducer

CCache build cache

```
hedin@laptop:~/.ccache/debugfs_bug$ CCACHE_DIR=~/.ccache/debugfs_bug/ ccache -s
Cacheable calls:   8238 / 8467 (97.30%)
  Hits:             371 / 8238 ( 4.50%)
    Direct:         369 /  371 (99.46%)
    Preprocessed:     2 /  371 ( 0.54%)
  Misses:          7867 / 8238 (95.50%)
Uncacheable calls:  229 / 8467 ( 2.70%)
Local storage:
  Cache size (GB):  1.5 / 20.0 ( 7.32%)
  Hits:             371 / 8238 ( 4.50%)
  Misses:          7867 / 8238 (95.50%)
```

Reproducer directory structure:

```
bug/
  build/ - kernel build output directory
    .config - syzbot generated config (put it here before
         build)
  patch/
    patch.diff - patch to apply
  reproduce/
    repro.c - syzbot generated C reproducer
```

Demo - patch source

```
hedin@laptop:~/prj/linux$ git diff
diff --git a/fs/debugfs/inode.c b/fs/debugfs/inode.c
index 532bd7c46baf..6a7b285a4cab 100644
--- a/fs/debugfs/inode.c
+++ b/fs/debugfs/inode.c
@@ -860,8 +860,10 @@ int __printf(2, 3) debugfs_change_name(struct dentry
     *dentry, const char *fmt, .
        rd.new_parent = rd.old_parent;
        rd.flags = RENAME_NOREPLACE;
        target = lookup_noperm_unlocked(&QSTR(new_name), rd.new_parent);
        - if (IS_ERR(target))
        - return PTR_ERR(target);
        + if (IS_ERR(target)) {
        + error = PTR_ERR(target);
        + goto out_free;
        + }
        error = start_renaming_two_dentries(&rd, dentry, target);
        if (error) {
@@ -881,6 +883,7 @@ int __printf(2, 3) debugfs_change_name(struct dentry
     *dentry, const char *fmt, .
 out:
        dput(rd.old_parent);
        dput(target);
+out_free:
        kfree_const(new_name);
        return error;
 }
```

Demo - Reproducer principle

Reproducer source code is big, instead it work description:

- Based on kmemleak (kernel's memory leak detector, similar to valgrind)
- Uses kmemleak control and info file `"/sys/kernel/debug/kmemleak"`
- Runs in an infinite loop, terminating when a memory leak is detected
  - Prints kernel stack trace reported by kmemleak

Demo - Reproduce the bug

Build reproducer:

```
$ gcc -o repro repro.c
```

Terminal 1: Run virtme-ng with built kernel

```
$ sudo vng --console \
    --run /home/hedin/fosdem/2026/syzbot/bug/build

hedin@virtme-ng:~/fosdem/2026/syzbot/bug/reproduce$ sudo ./repro
executing program
executing program
BUG: memory leak
unreferenced object 0xffff8880211891d8 (size 8):
  comm "repro", pid 10177, jiffies 4294942828
  hex dump (first 8 bytes):
    2e 00 00 00 00 00 00 00                          ........
backtrace (crc ecfc7064):
    __kmalloc_node_track_caller_noprof+0x3cb/0x670
    kstrdup+0x3c/0x80
    kstrdup_const+0x63/0x80
    kvasprintf_const+0xca/0x110
    ..............................................
```

Terminal 2: Connect via vsock

```
$ vng --console-client

hedin@virtme-ng:~/fosdem/2026/syzbot/bug/reproduce$ sudo dmesg -kw
[ 61.275543] kmemleak:  1 new suspected memory leaks (see /sys/kernel/debug/kmemleak)
```

Demo - Fix the bug

Apply patch

```
$ git apply ~/fosdem/2026/syzbot/bug/patch/patch.diff
```

Build kernel:

```
# Build config based on syzbot generated
$ KBUILD_BUILD_TIMESTAMP='' CCACHE_DIR=~/.ccache/debugfs_bug/ \
  time make CC="ccache gcc" \
  O=/home/hedin/fosdem/2026/syzbot/bug/build \
  oldconfig -j$(nproc)

# Build Kernel
$ KBUILD_BUILD_TIMESTAMP='' CCACHE_DIR=~/.ccache/debugfs_bug/ \
  time make CC="ccache gcc" \
  O=/home/hedin/fosdem/2026/syzbot/bug/build -j$(nproc)
```

Terminal:

```
hedin@virtme-ng:~/fosdem/2026/syzbot/bug/reproduce$ sudo ./repro
executing program
executing program
executing program
executing program
executing program
executing program
```

Where virtme-ng Helps (and Where It Doesn't)

Inefficient workflows

- Testing many patches across many kernel versions
- Frequent make clean / full rebuilds
- Switching between unrelated kernel trees (unless you have per-kernel CCACHE dir)

Efficient workflows

- Testing multiple patches on the same kernel version
- Iterating on a bug fix

Thank You

Questions?

References

- virtme-ng github repo
https://github.com/arighi/virtme-ng
- syzkaller
https://github.com/google/syzkaller
- syzkaller bot
https://syzkaller.appspot.com/upstream
- Kernel Recipes 2024 - virtme-ng
https://www.youtube.com/watch?v=pw0kA9w3kUo
- Mentorship Session: Speeding Up Kernel Development With virtme-ng
https://www.youtube.com/watch?v=ZgMLGM2UazY
- AI-assisted Linux kernel patch testing (opencode + virtme-ng + local LLM)
https://asciinema.org/a/763692
- Reference to a bug that I will reproduce
https://syzkaller.appspot.com/bug?extid=3d7ca9c802c547f8550a

- asciinema recording of bug reproducing
  https://asciinema.org/a/767681
- asciinema recording of bug fixing
  https://asciinema.org/a/767683