Linux Plumbers Conference 2023
Richmond, VA

Syzbot: 7 years of continuous kernel fuzzing

Aleksandr Nogikh <nogikh@google.com>
Google

Agenda

- Introduction
- Ignored vs Addressed Findings
- 2023 Updates
- Bug Analysis
- Controversial Topics
- Questions / Discussion

Syzbot

- **syzkaller** (coverage-guided kernel fuzzer) appeared in **2015.**
  - Syzkaller is a standalone application.
- **syzbot** has begun to report kernel findings to LKML in **2017.**
  - Syzbot is a continuous kernel build / fuzz / report aggregation system.
  - Syzbot uses **syzkaller** for the actual fuzzing.
- **~17k** findings detected and **~6k** reported to LKML.
- **3400+** Linux kernel commits directly mention syzbot.
  - Syzbot's web dashboard records **4800+** resolved findings.

Syzbot Reports

```
From: syzbot @ 2023-09-25 18:58 UTC (permalink / raw)

Hello,

syzbot found the following issue on:

HEAD commit:    42dc814987c1 Merge tag 'media/v6.6-2' of git://git.kernel...
git tree:       upstream
console output: https://syzkaller.appspot.com/x/log.txt?x=153c42d4680000
kernel config:  https://syzkaller.appspot.com/x/.config?x=e4ca82a1bedd37e4
dashboard link: https://syzkaller.appspot.com/bug?extid=53034ab3f4d670ca496b
compiler:       Debian clang version 15.0.6, GNU ld (GNU Binutils for Debian) 2.40

< ... >
```

+ Reproducers / Downloadable files / Stack traces

Web Dashboard

https://syzkaller.appspot.com

< ... >

open (813):

Title	Repro	Cause bisect	Fix bisect	Count	Last	Reported	Discussions
UBSAN: shift-out-of-bounds in radix_tree_next_chunk [kernel]				1	4d11h	10h59m	0 [10h59m]
general protection fault in tls_merge_open_record [net]	syz			5	14h53m	14h52m	0 [14h52m]
general protection fault in __hugetlb_zap_begin [mm]	C	done		11	1h53m	1d03h	0 [1d03h]
general protection fault in hugetlb_vma_lock_write [mm]	C	done		12	4h38m	1d11h	0 [1d11h]
possible deadlock in indx_read [ntfs3]				1	5d22h	1d22h	0 [1d22h]

Web Dashboard (2)

general protection fault in tls_merge_open_record

Status: upstream: reported syz repro on 2023/10/30 05:52
Subsystems: net
[Documentation on labels]
Reported-by: syzbot+40d43509a099ea756317@syzkaller.appspotmail.com
First crash: 64d, last: 15h17m

▶ Discussions (1)

Sample crash report:

```
general protection fault, probably for non-canonical address 0xdffffc0000000001: 0000 [#1] PREEMPT SMP KASAN
KASAN: null-ptr-deref in range [0x0000000000000008-0x000000000000000f]
CPU: 1 PID: 12569 Comm: syz-executor.0 Not tainted 6.6.0-rc7-next-20231027-syzkaller #0
Hardware name: Google Google Compute Engine/Google Compute Engine, BIOS Google 10/09/2023
RIP: 0010:_compound_head include/linux/page-flags.h:247 [inline]
RIP: 0010:put_page include/linux/mm.h:1544 [inline]
```

< ... >

Crashes (5):

Time	Kernel	Commit	Syzkaller	Config	Log	Report	Syz repro	C repro	VM info	Assets (help?)	Manager
2023/10/30 05:51	linux-next	66f1e1ea3548	3c418d72	.config	console log	report	syz			[disk image] [vmlinux] [kernel image]	ci-upstream-linux-next-kasa
2023/09/09 06:21	upstream	a48fa7efaf11	6654cf89	.config	console log	report			info	[disk image] [vmlinux] [kernel image]	ci-upstream-kasan-gce-selir
2023/08/26 22:16	upstream	7d2f353b2682	03d9c195	.config	console log	report			info	[disk image] [vmlinux] [kernel image]	ci-upstream-kasan-gce
2023/10/19 23:55	net-next	b91f2e13c972	42e1d524	.config	console log	report			info	[disk image] [vmlinux] [kernel image]	ci-upstream-net-kasan-gce
2023/10/29 20:25	linux-next	66f1e1ea3548	3c418d72	.config	console log	report			info	[disk image] [vmlinux] [kernel image]	ci-upstream-linux-next-kasa

Mainline Linux Kernel Fuzzing

Covered targets:

- GCE/x86_64
- GCE/arm64
- qemu/x86_64 (KVM)
- qemu/arm32 (emu)
- qemu/arm64 (emu)
- qemu/RISC-V (emu)

Covered trees:

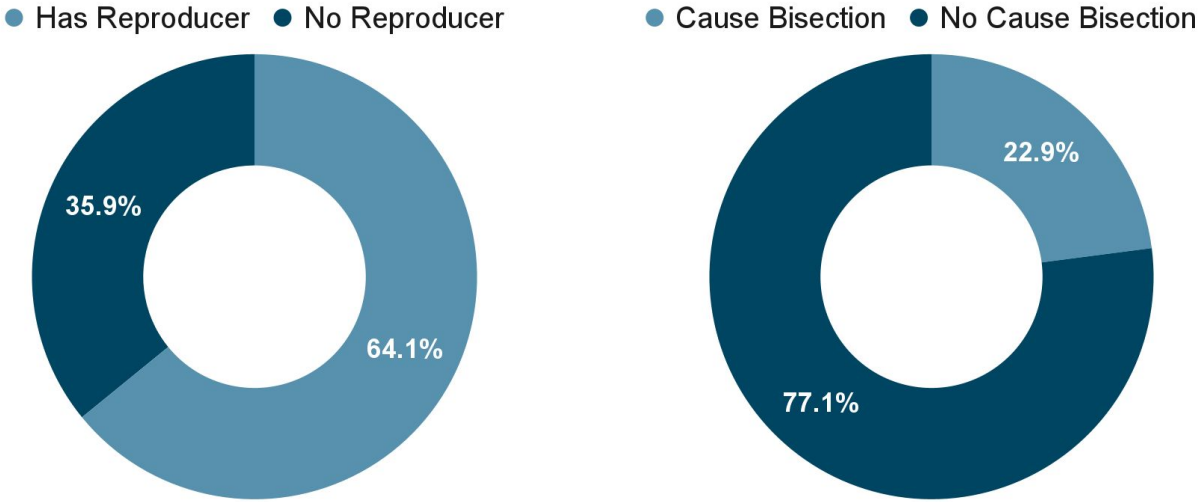
- torvalds/master
- linux-next/master
- bpf/master
- bpf-next/master
- other fuzzed mainline trees

Linux kernel is fuzzed on **25** syzkaller instances using **~150-200** VMs in total.

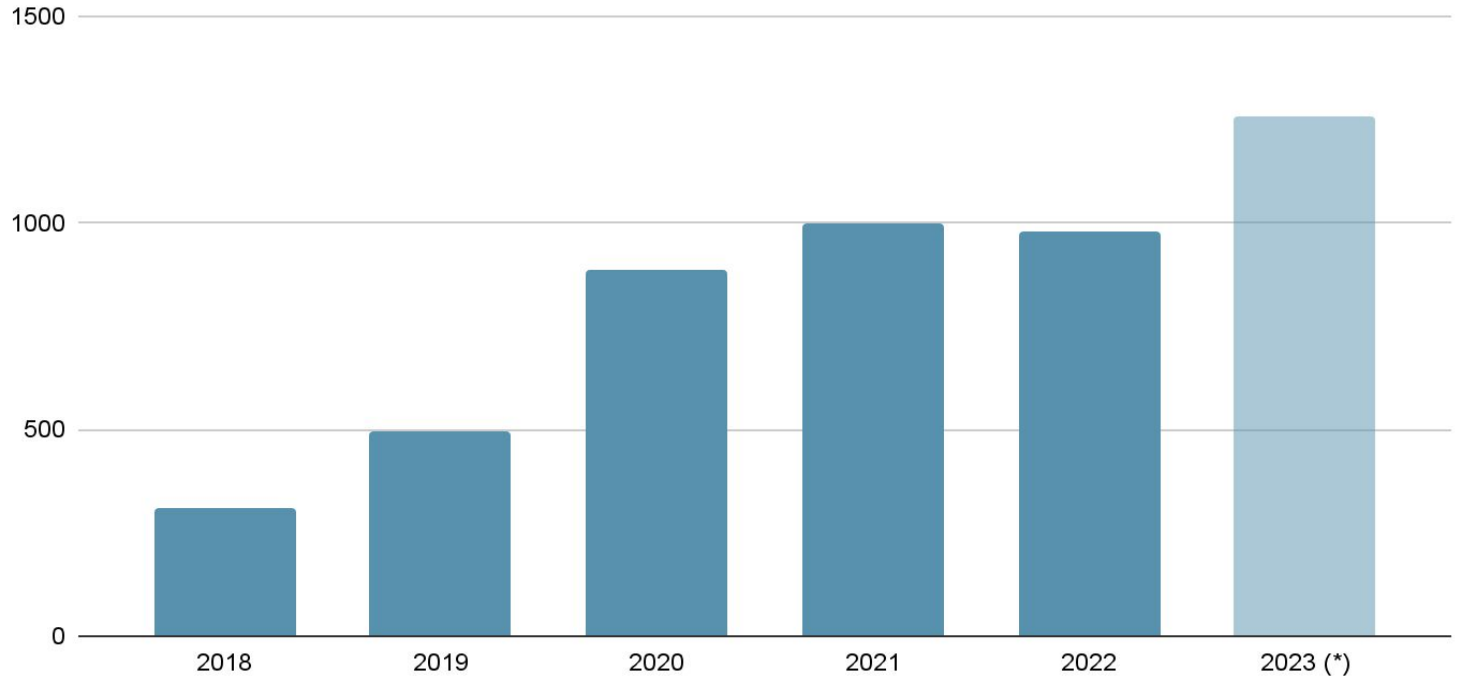
Yearly Figures

(*) Commits in the "torvalds" tree that mention syzbot or syzkaller.appspot.com.

Reported Findings (2020-2023)



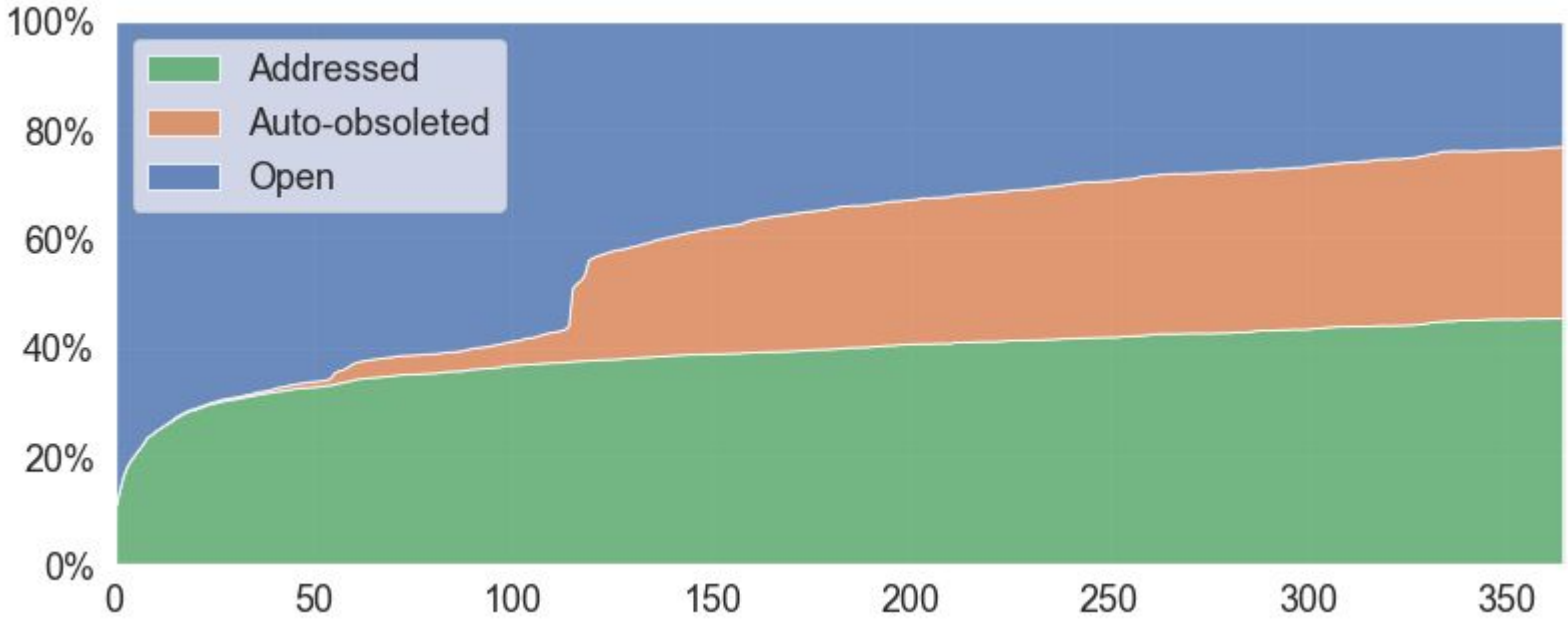
Patch Testing Requests



(*) Extrapolation based on the data 01/2023-10/2023.

Ignored vs Addressed Findings

Reported Findings: Status Distribution



Days after reporting to LKML

Report Factor Importance

F(Report Factors) = { **True** if the report was addressed within 45 days(*)

False otherwise

Q: What report factors are most important?

(*) **45 days** is a convenient figure:

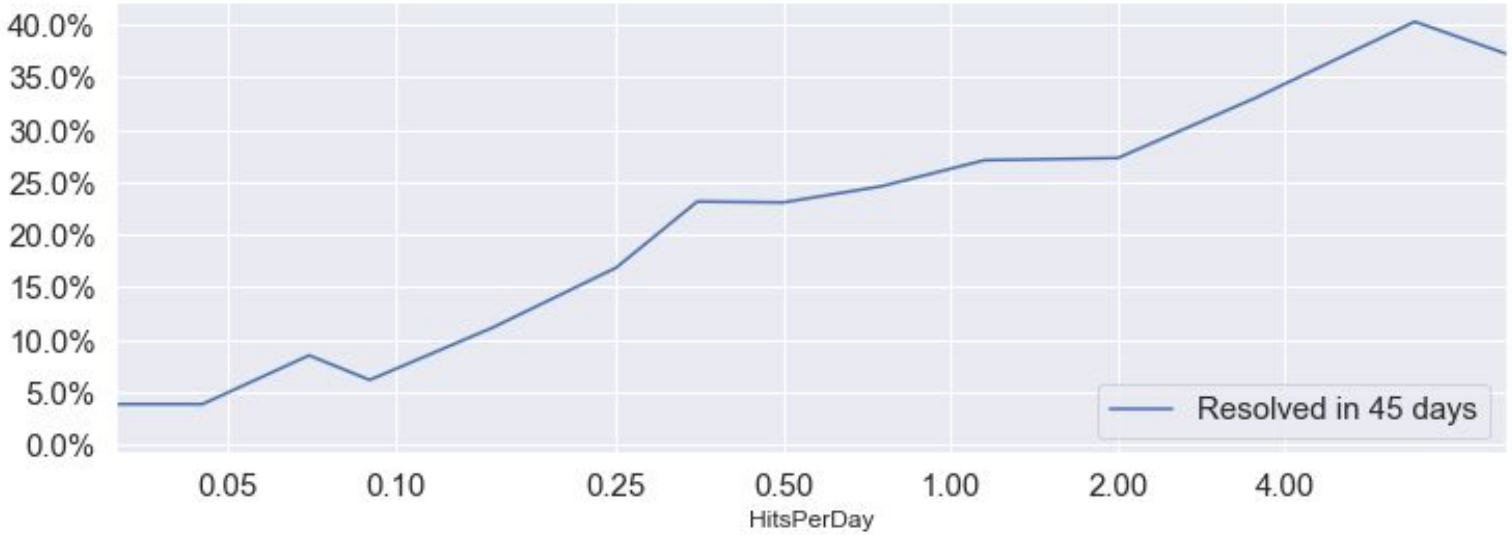
- **72%** reports that are ever addressed are addressed within **45 days**.
- Automatic bug obsoletion comes into effect later.

Features Importance (per Mutual Information)

- Affected kernel subsystem.
- Average recorded Hits/Day *(bucketed).*
- Cause Bisection present.
- Report type (KASAN, BUG, WARNING, lockdep, etc.).
- Report month / week day / hour *(bucketed).*
- Reproducer present.

Dataset: syzbot reports to public mailing lists 2020-2023.

Effect of Average Hits/Day on %% addressed in 45 days



Yes, it's a surprisingly strong correlation.

No, it's not explainable by higher repro/cause bisect success rates.

Effects of Repro and Cause Bisection

	Reproducer: NO	**Reproducer: YES**
Cause Bisection: NO	**14%** addressed in 45 days	**19%** addressed in 45 days
Cause Bisection: YES	*impossible*	**39%** addressed in 45 days

Effect of Report Type

Some examples.

Report Type	Addressed in 45 days
UBSAN	30%
general protection fault	27%
KASAN	20%
WARNING	20%
lockdep	20%
INFO: task hung	10%

2023 Updates

Cause Bisections

More bisections:

2023Q3 findings with a reproducer: ~40% have cause bisection

2022 findings with a reproducer: ~20% have cause bisection

Better precision (see next slide)

Cause Bisections: Challenges

- Many kernel revisions do not build/boot with syzbot config.
  - We cherry-pick a number of commits to address known build/boot failures.
  - **New:** kernel config is partially minimized before bisection.
- Bug reproducers are not always reliable.
  - **New:** syzbot estimates accumulated error probability and applies a threshold.
  - Stochastic git bisections could really help here.
- Single reproducer might trigger several unrelated bugs.
  - **New:** syzbot drops unnecessary instrumentation and ignores unrelated crashes. **But that's not a 100% remedy :(**
- Bisecting by reproducer points not to the culprit, but to the commit that surfaced the bug.
  - **Could it be ever resolved automatically?**

LKML Discussions Monitoring

On each per-report page on the Web Dashboard (example)

▼ Discussions (3)

Title	Replies (including bot)	Last reply
[PATCH v2] net/tls: Fix slab-use-after-free in tls_encrypt_done	1 (1)	2023/10/17 16:22
[PATCH] net/tls: Fix slab-use-after-free in tls_encrypt_done	5 (5)	2023/10/17 11:49
[syzbot] [net?] KASAN: slab-use-after-free Read in tls_encrypt_done	0 (1)	2023/09/29 18:43

In every list of open findings

💬 0 [11d]	no comments
PATCH [7d02h]	has a patch (patch candidate) that was last commented 7 days ago
💬 1 [12d]	one user comment 12 days ago

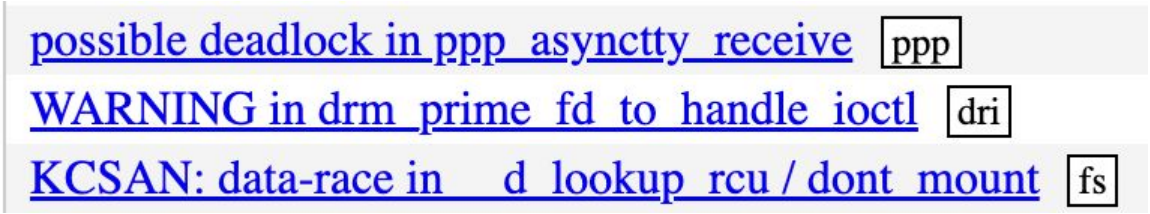
Subsystem Labels

https://syzkaller.appspot.com/upstream/subsystems

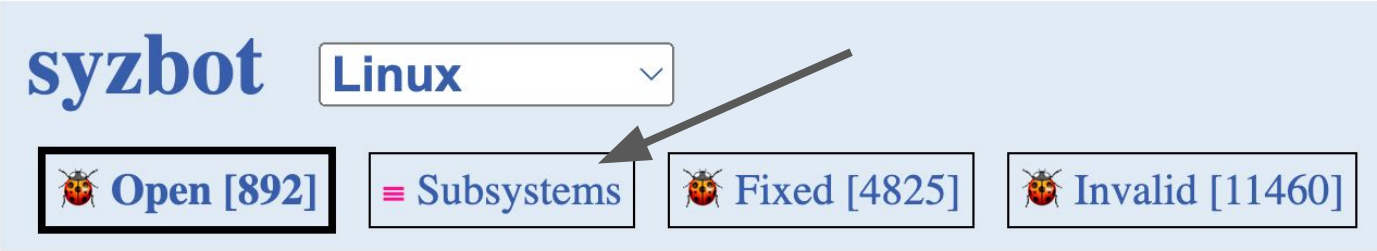
Email Subjects:

```
* [syzbot] [wireguard?] WARNING in kthread_unpark
@ 2023-10-08 15:27 syzbot
```

Web Dashboard:



Subsystem Pages



The list of subsystems

https://syzkaller.appspot.com/upstream/subsystems

Name	List(s)	Open	Fixed
acpi	linux-acpi@vger.kernel.org	1	2
afs	linux-afs@lists.infradead.org	1	40
alsa	alsa-devel@alsa-project.org	1	106

Subsystem Pages (2)

bluetooth subsystem

List(s): linux-bluetooth@vger.kernel.org
Maintainer(s): johan.hedberg@gmail.com, luiz.dentz@gmail.com, marcel@holtmann.org
Fixed bugs: 58
Parent subsystem(s): kernel (33)

open (26):

Title	Repro	Cause bisect	Fix bisect	Count	Last	Reported
KASAN: slab-use-after-free Read in sco_chan_del bluetooth				1	6d03h	3h55m
KASAN: slab-use-after-free Read in release_sock bluetooth				1	10d	6d03h
possible deadlock in hci_rfkill_set_block bluetooth	C	done		442	1h24m	7d16h
memory leak in prepare_creds (4) bluetooth	syz			1	21d	32d
possible deadlock in hci_dev_do_close bluetooth	C	done		1799	19m	36d
KASAN: null-ptr-deref Write in l2cap_sock_suspend_cb bluetooth				1	47d	40d
general protection fault in lock_sock_nested bluetooth	C	done		47	17h49m	50d

Subsystems: List Construction

- We needed a **sensibly-sized list** of **short names** to be used as tags.
- **MAINTAINERS** file contains very relevant information, but:
  - **Too many entries** (>2700 as of v6.6).
  - **Too long titles** that cannot be used as tags.
- For syzbot, we grouped **MAINTAINERS** records by mailing lists, e.g.
  - *kvm@vger.kernel.org* -> kvm
  - *linux-serial@vger.kernel.org* -> serial
  - Plus a handful of exceptions, of course.
- Result: **238** subsystems (as of October 2023).

Subsystems: Classification

We auto-generate the list of rules that map every subsystem to:

- Path regexps (taken from **MAINTAINERS**).
  - This is to be matched agains stack traces.
- Relevant calls from reproducers (**manually crafted**).

Overall algorithm is straightforward:

Take X top crash reports for every bug, extract subsystems for every crash, aggregate the results.

(Details are omitted, look here to find out more)

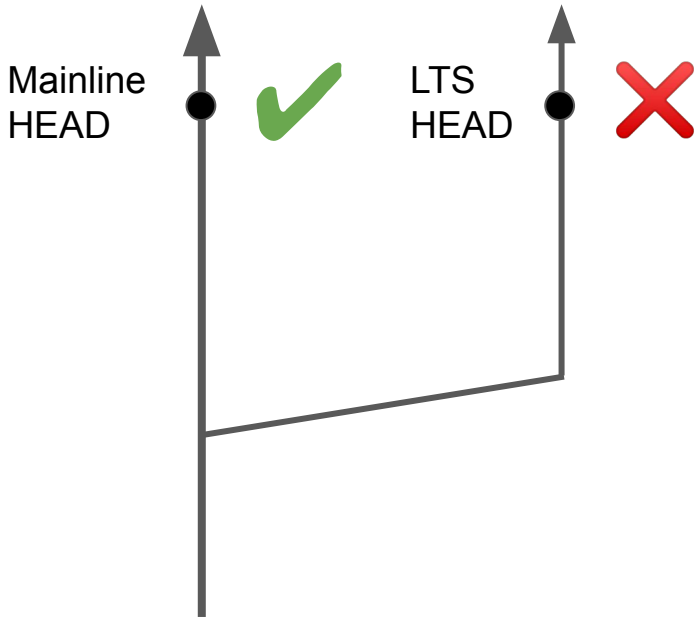
Subsystems: Limitations

- **Sometimes there are false positives**, it's affected by other error-prone functionality:
  - Unrelated crashes grouped together.
  - Stack traces may be misleading.
    - They span over multiple different subsystems.
    - They don't include the actual guilty frame.
- **We periodically recalculate subsystem labels** as we collect more crashes.
  - It's especially problematic in mistakenly glued reports.
  - But no labels updated via `#set subsystem` are overwritten.
- **Still, in the majority of cases**, the precision look good.

The subsystems list and the classifications rules are there to be adjusted to your needs.
Please feel free to contact us at syzkaller@googlegroups.com.

Bug Analysis

Kernel Bug Presence (example)



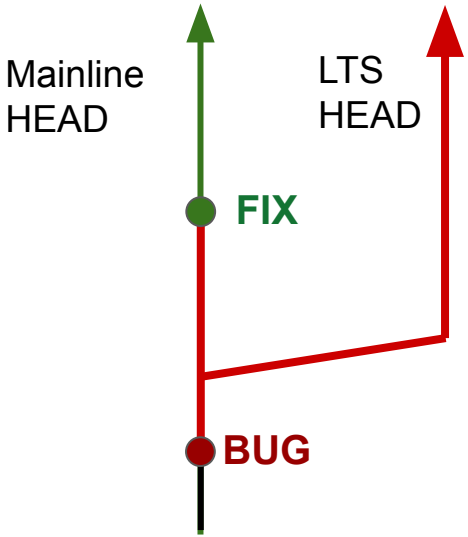
A bug in an LTS kernel is found.

We run reproducer on two trees:

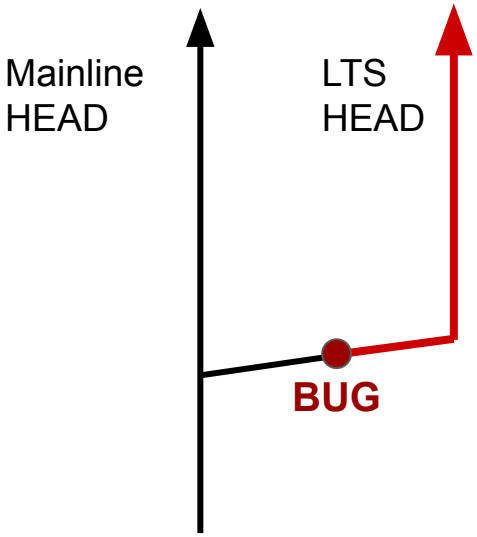
- HEAD of **LTS**: crashes.
- HEAD of **Mainline**: doesn't crash.

What does it mean?

Kernel Bug Presence (example)



Missing backport from Mainline



Problem that never existed in Mainline

Some corner cases:

- Bug reproducer is unreliable.
- Reproducer triggers several bugs.

Let's assume the chances are not very high.

LTS-Only Bugs on Syzbot

Syzbot performs this analysis for two Linux LTS versions:

Data as of October 2023

5.15	421 open bugs	**96 open bugs are LTS-only (~23%)**	192 open bugs are also in Mainline (~45%)	No decision for 133 bugs (32%)
6.1	388 open bugs	**68 open bugs are LTS-only (~17%)**	192 open bugs are also in Mainline (~49%)	No decision for 128 bugs (34%)

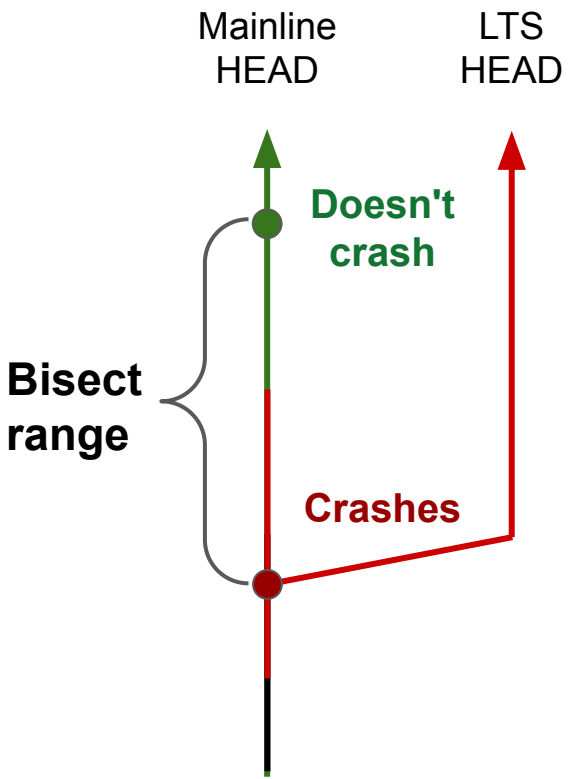
These likely have non-backported fixes

Missing Backports

- Bug reproduces on the merge base between Mainline and LTS
- Bug does not reproduce on HEAD of Mainline

We can perform a bisection to find the non-backported fixing commit.

With improvements to the bisection process, we can even expect reasonably good results.



Missing Backports: Current Results

https://syzkaller.appspot.com/upstream/backports

Manual analysis (as of October 2023):

	Linux 5.15 LTS	**Linux 6.1 LTS**
Total Found	32	32
Correct	21 (65%)	26 (81%)
No `Fixes:` tag	19 of 21 (90%)	21 of 26 (81%)

What are those commits?

Among the correctly identified backport candidates:

1. Actual bug fixes: **30 of 47 (~64%)**
2. Refactorings and optimizations: **9 of 47 (~20%)**
3. Removed or fixed an invalid code assertion: **5 of 47 (~10%)**
4. Kernel feature deprecations: **3 of 47 (~6%)**

Controversial Topics

"Please don't fuzz/report bugs in XYZ"

Conflicting Points:

- There's no point in sending reports that are
  - Unlikely to be ever addressed,
  - Not perceived as bugs by the kernel development community.
- If the code is in the kernel and compiled in by many Linux distributions, is it correct to ignore problems in it?

Compromise Solution [currently being implemented]**:**

Such findings are **not reported** via email, but displayed on the web dashboard and labeled with a special tag.

Low severity and low priority reports

Complaints:

- Syzbot reports shallow problems.
- Syzbot exercises code paths never meant for real-world use.

New:

- Specify priority and filter findings by priority on the web dashboard:
  `#syz set prio: low`
- Exclude a finding from monthly reporting:
  `#syz set no-reminders`

For repetitive cases, please contact us at syzkaller@googlegroups.com

Low severity and low priority reports (2)

syzkaller (as a fuzzing tool) would trigger more interesting problems if:

- There are more descriptions of the target subsystem's interface.
  - Descriptions let it generate more meaningful programs that go deeper into the code.
- There are no crashes fuzzing stumbles on at the very beginning.
- The kernel code is using assertions with extra care.

Name	Last active	Uptime	Corpus	Coverage
ci-qemu-upstream	now	4h18m	41648	605503

Maintainer Burnout

Complaint:

syzbot contributes to the overload of Linux kernel maintainers.

What can syzbot do to improve the situation from its side?

One option could be to "shift-left" kernel fuzzing (i.e. fuzz also incoming patches).

- More bugs are discovered and fixed before merging => less stress for maintainers later.
- The "lightweight" approach: apply incoming patch, build an instrumented kernel, run syzbot's corpus (40-50k programs).
  - An efficiency evaluation must be performed first.
  - Can it be done on existing/developed kernel CIs?

False Positives

- Appear in multiple places.
  - Invalid bisection results.
  - Incorrectly inferred subsystems.
  - Incorrectly merged reports.
  - False positive reports.
  - Not fully minimized reproducers.
- We try to focus on eliminating whole classes of false positives.
  - **Individual ones are unfortunately always to expect**.
- Some may only be addressed with changes to both syzbot and the kernel.
- If you have any specific ideas/suggestions, please let us know.

False Positive Reports

- Kernel bugs are detected by the **kernel** itself, **syzkaller** just stress-tests it and parses reports from the serial console/dmesg.
  - Improvements to kernel's bug detection benefit all, not just syzbot/syzkaller.
  - Improvements may include e.g. better sanitizers and proper use of assertions.
- Kernel configs that disable potentially dangerous functionality are of great help for fuzzing. Some examples include:
  - CONFIG_DEVMEM=n that disables /dev/mem.
  - The block: Add config option to not allow writing to mounted devices series by Jan Kara will soon help eliminate a big class of undesired filesystem reports.

Birds of a Feather Session

Topic:

How to make syzbot reports easier to debug?

Wed 15/11, 10:15 AM - 11:00 AM

Linux Plumbers Conference 2023
Richmond, VA

Syzbot: 7 years of continuous kernel fuzzing

Aleksandr Nogikh <nogikh@google.com>
Google