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

<table>
<thead>
<tr>
<th>Title</th>
<th>Repro</th>
<th>Cause bisect</th>
<th>Fix bisect</th>
<th>Count</th>
<th>Last</th>
<th>Reported</th>
<th>Discussions</th>
</tr>
</thead>
<tbody>
<tr>
<td>UBSAN: shift-out-of-bounds in radix_tree_next_chunk</td>
<td>kernel</td>
<td>syz</td>
<td>done</td>
<td>1</td>
<td>4d11h</td>
<td>10h59m</td>
<td>0 [10h59m]</td>
</tr>
<tr>
<td>general protection fault in tls_merge_open_record</td>
<td>net</td>
<td>syz</td>
<td>done</td>
<td>5</td>
<td>14h53m</td>
<td>14h52m</td>
<td>0 [14h52m]</td>
</tr>
<tr>
<td>general protection fault in hugetlb_zap_begin</td>
<td>mm</td>
<td>C</td>
<td>done</td>
<td>11</td>
<td>1h53m</td>
<td>1d03h</td>
<td>0 [1d03h]</td>
</tr>
<tr>
<td>general protection fault in hugetlb_vma_lock_write</td>
<td>mm</td>
<td>C</td>
<td>done</td>
<td>12</td>
<td>4h38m</td>
<td>1d11h</td>
<td>0 [1d11h]</td>
</tr>
<tr>
<td>possible deadlock in indx_read</td>
<td>mfs3</td>
<td></td>
<td></td>
<td>1</td>
<td>5d22h</td>
<td>1d22h</td>
<td>0 [1d22h]</td>
</tr>
</tbody>
</table>
Web Dashboard (2)

general protection fault in tls_merge_open_record

Status: ![upstream reported syz repro on 2023/10/30 05:52](web_dashboard)
Subsystems: ![net](web_dashboard)

[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 0xffffffff: 0000 [0] PREEMPT SMP KASAN
KASAN: null-ptr-de in range [0x0000000000000000-0x000000000000000f]
CPU: 1 PID: 12569 Comm: syz-executor.0 Not tainted 6.6.0-rc7-next-20231027-syzkaller #0
Hardware name: 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]

< ... >
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)

- Has Reproducer: 35.9%
- No Reproducer: 64.1%

- Cause Bisection: 22.9%
- No Cause Bisection: 77.1%
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

\[ F(\text{Report Factors}) = \begin{cases} 
  \text{True} & \text{if the report was addressed within } 45 \text{ days} (*) \\
  \text{False} & \text{otherwise}
\end{cases} \]

Q: What report factors are most important?

(*) \textbf{45 days} is a convenient figure:

- 72\% reports that are ever addressed are addressed within \textbf{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

<table>
<thead>
<tr>
<th>Reproducer: NO</th>
<th>Reproducer: YES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause Bisection: NO</strong></td>
<td>14% addressed in 45 days</td>
</tr>
<tr>
<td><strong>Cause Bisection: YES</strong></td>
<td>impossible</td>
</tr>
</tbody>
</table>
Effect of Report Type

Some examples.

<table>
<thead>
<tr>
<th>Report Type</th>
<th>Addressed in 45 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>UBSAN</td>
<td>30%</td>
</tr>
<tr>
<td>general protection fault</td>
<td>27%</td>
</tr>
<tr>
<td>KASAN</td>
<td>20%</td>
</tr>
<tr>
<td>WARNING</td>
<td>20%</td>
</tr>
<tr>
<td>lockdep</td>
<td>20%</td>
</tr>
<tr>
<td>INFO: task hung</td>
<td>10%</td>
</tr>
</tbody>
</table>
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](#))

In every list of open findings

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

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:

possible deadlock in ppp_asyncTTY_receive

WARNING in drm_prime_fd_to_handle_ioctl

KCSAN: data-race in d_lookup_rcu / dont_mount
Subsystem Pages

The list of subsystems

<table>
<thead>
<tr>
<th>Name</th>
<th>List(s)</th>
<th>Open</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>acpi</td>
<td><a href="mailto:linux-acpi@vger.kernel.org">linux-acpi@vger.kernel.org</a></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>afs</td>
<td><a href="mailto:linux-afs@lists.infradead.org">linux-afs@lists.infradead.org</a></td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>alsa</td>
<td><a href="mailto:alsa-devel@alsa-project.org">alsa-devel@alsa-project.org</a></td>
<td>1</td>
<td>106</td>
</tr>
</tbody>
</table>
### 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)

<table>
<thead>
<tr>
<th>Title</th>
<th>Repro</th>
<th>Cause bisect</th>
<th>Fix bisect</th>
<th>Count</th>
<th>Last</th>
<th>Reported</th>
</tr>
</thead>
<tbody>
<tr>
<td>KASAN: slab-use-after-free Read in sctp_chan_del</td>
<td>bluetooth</td>
<td></td>
<td></td>
<td>1</td>
<td>6d03h</td>
<td>3h55m</td>
</tr>
<tr>
<td>KASAN: slab-use-after-free Read in release_sock</td>
<td>bluetooth</td>
<td></td>
<td></td>
<td>1</td>
<td>10d</td>
<td>6d03h</td>
</tr>
<tr>
<td>possible deadlock in hci_rkill_set_block</td>
<td>bluetooth</td>
<td>C</td>
<td>done</td>
<td>442</td>
<td>1h24m</td>
<td>7d16h</td>
</tr>
<tr>
<td>memory leak in prepare_creds</td>
<td>bluetooth</td>
<td>syz</td>
<td></td>
<td>1</td>
<td>21d</td>
<td>32d</td>
</tr>
<tr>
<td>possible deadlock in hci_dev_do_close</td>
<td>bluetooth</td>
<td>C</td>
<td>done</td>
<td>1799</td>
<td>19m</td>
<td>36d</td>
</tr>
<tr>
<td>KASAN: null-ptr-deref Write in l2cap_sock_suspend_cb</td>
<td>bluetooth</td>
<td></td>
<td></td>
<td>1</td>
<td>47d</td>
<td>40d</td>
</tr>
<tr>
<td>general protection fault in lock_sock_nested</td>
<td>bluetooth</td>
<td>C</td>
<td>done</td>
<td>47</td>
<td>17h49m</td>
<td>50d</td>
</tr>
</tbody>
</table>
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 against 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
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*

<table>
<thead>
<tr>
<th>Version</th>
<th>Total Bugs</th>
<th>LTS-Only Bugs (#{percentage})</th>
<th>Mainline Bugs (#{percentage})</th>
<th>No Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.15</td>
<td>421</td>
<td>96 (~23%)</td>
<td>192 (~45%)</td>
<td>133 (32%)</td>
</tr>
<tr>
<td>6.1</td>
<td>388</td>
<td>68 (~17%)</td>
<td>192 (~49%)</td>
<td>128 (34%)</td>
</tr>
</tbody>
</table>

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):

<table>
<thead>
<tr>
<th></th>
<th>Linux 5.15 LTS</th>
<th>Linux 6.1 LTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Found</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Correct</td>
<td>21 (65%)</td>
<td>26 (81%)</td>
</tr>
<tr>
<td>No <code>Fixes:</code> tag</td>
<td>19 of 21 (90%)</td>
<td>21 of 26 (81%)</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Last active</th>
<th>Uptime</th>
<th>Corpus</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci-qemu-upstream</td>
<td>now</td>
<td>4h18m</td>
<td>41648</td>
<td>605503</td>
</tr>
</tbody>
</table>
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