Maintaining results from static analysis collaboratively?

Lukas Bulwahn
Can we, the kernel community, maintain results from static analysis tools collaboratively?

Do we want to...?

and how do we ...?
Static Analysis Tools in the Kernel Development

- extended compiler warnings (gcc, clang)
- sparse
- smatch
- coccinelle
- checkpatch.pl
- coverity
- clang-tidy
- … and probably a few more

Assume any tool of your choice...
Different technical difficulties might arise with the different tools, but the general question remains the same.
Some Definitions

There are different interpretations of the term “False Positives”...
... so let us avoid the term... and use:

**True Tool Finding (True Positive, Type A):**
Tool based on some heuristics reports conditions that describe execution paths that could really happen in some scenario and ultimately leads to an non-intended behavior of a kernel functionality (a bug).

**False Tool Finding (False Positive, Type A):**
Tool based on some heuristic certain flow reports conditions that describe execution paths that can never happen or does not ultimately lead to an non-intended behavior of a kernel functionality (a bug).

**True Tool-Induced Change (True Positive, Type B):**
A finding reported by a static analysis tool that can lead to a developer making a code change meeting the requirements to be included (accepted by the maintainer).

**False Tool-Induced Change (False Positive, Type B):**
A finding reported by a static analysis tool for which any derived code change does not meet the requirements to be included; the maintainer rejects to change the source code in any way.
Generally Executed Workflow
(Assumption for later discussion)

Conclusion: Despite all reasonable effort, there are cases where a static tool continues to report a finding and we must accept that we do not change the source or the tool.
Status of tool findings on current kernel

Despite all reasonable effort, there are cases where a static tool continues to report a finding and we must accept that we do not change the source or the tool.

Is this a real situation in the current kernel development?

~22,000 warnings and errors with sparse on v5.9-rc1 allyesconfig
~4,700 warnings and errors with coccinelle on v5.9-rc1
...

Discussions with tool authors, long-term kernel janitors and multiple smaller evaluations suggest:
Most of those findings cannot be (economically) silenced or patches are not accepted.
Core Problem and Core Question

Core Problem for a larger group of distributed tool users on one code base:
With increasing analysis contribution efforts, we must accept that we do not change the source or the tool for an increasing ratio and amount of tool findings.

New tool users first need to go through all already existing tool findings that cannot be addressed.

Discussion:
- Is there interest to find a collaborative solution to this problem?
- How could this solution work?
  - Organizationally?
  - Technically?
Value and Interest I.

Do you think the use of static analysis tools deserve more attention within the kernel community?
A. Yes
B. No

Is the use and reporting of static analysis tools already a well-organized community effort?
A. Yes, no improvements needed.
B. Yes, but we need to improve.
C. No, just single individual efforts.
D. No, and we should not waste time on that.
Value and Interest II.

Are you already involved with the use of static analysis tools?
A. Yes, I am continuously tracking the findings and reacting.
B. Yes, I tried it once and react to reported findings.
C. No, I tried it once and gave up.
D. No, I considered it to difficult to set up and run.
E. No, I did know how to engage.

Would you engage and participate in more collaboratively organized effort of static analysis tool use and triage?
A. Yes, I would even go the extra mile.
B. Yes, but it needs to fit into my current working mode.
C. No
D. No, please go away.
Value and Interest III.

If you would have 100 coins, how would you distribute efforts for improvements on...

(A = 0, B = 10, C = 25, D = 50, E = 90, F = 100)

1. ... testing, i.e., more tests/test suites?
2. ... dynamic analysis, i.e., fix all syzbot findings?
3. ... better kernel documentation?
4. ... addressing findings of static analysis?
5. ... anything else in the kernel development?
Organisational Aspects

We have file/directory/subsystem-specific configuration of what tools shall report.

Where and how to store?
Who maintains that information?
How to collaborate?

We have findings from tools that are continued to reported across versions.

Where and how to store?
Who maintains that information?
How to collaborate?

How to ensure confidence in these assessments?
What if multiple assessors disagree with their assessments?
Early analysis does not resolve the issue

When to run/inform about static analysis tool reports?

- **patch creation**
  - not enforceable.
  - compute intensive on dev. clients

- **patch inclusion (linux-next)**
  - already late for author interaction.
  - setup simpler.

- **kernel release**
  - already introduces churn of backporting decision.

- **patch submission**
  - enforceable.
  - partly done. little infrastructure available.

- **kernel rc’s**
  - most observable activity, IMHO.

Still a database of records on the assessments of the static analysis findings are required, independent of where static analysis tools are invoked during the development process.
Technical Aspects and Solutions

Envisioned User Interface for the different phases:

- Patch creation
- Discussion on mailing list
- While tracking various trees (linux-next, mainline, stable releases)

What are the preferred interfaces?

What are the core requirements for such a technical system?
Backup
Some First Experiments with one Technical Solution
Value stored to 'nr' during its initialization is never read

Value stored to 'nr' during its initialization is never read