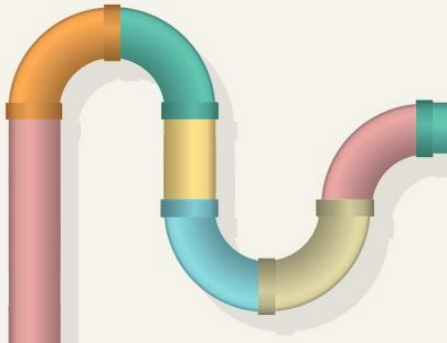
Maintaining results from static analysis collaboratively?

Lukas Bulwahn



LINUX
PLUMBERS
CONFERENCE

August 24-28, 2020

Core Question for this Discussion

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

Do we want to...?

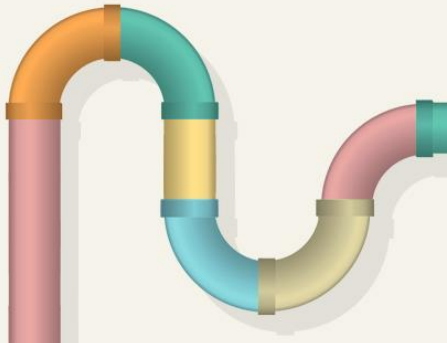
and how do we ...?

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

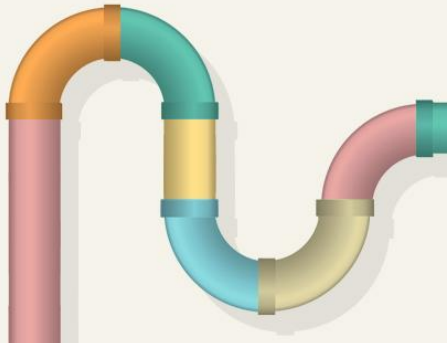
There are different interpretations of the term “False Positives”...
… so let us avoid the term… and use:

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.

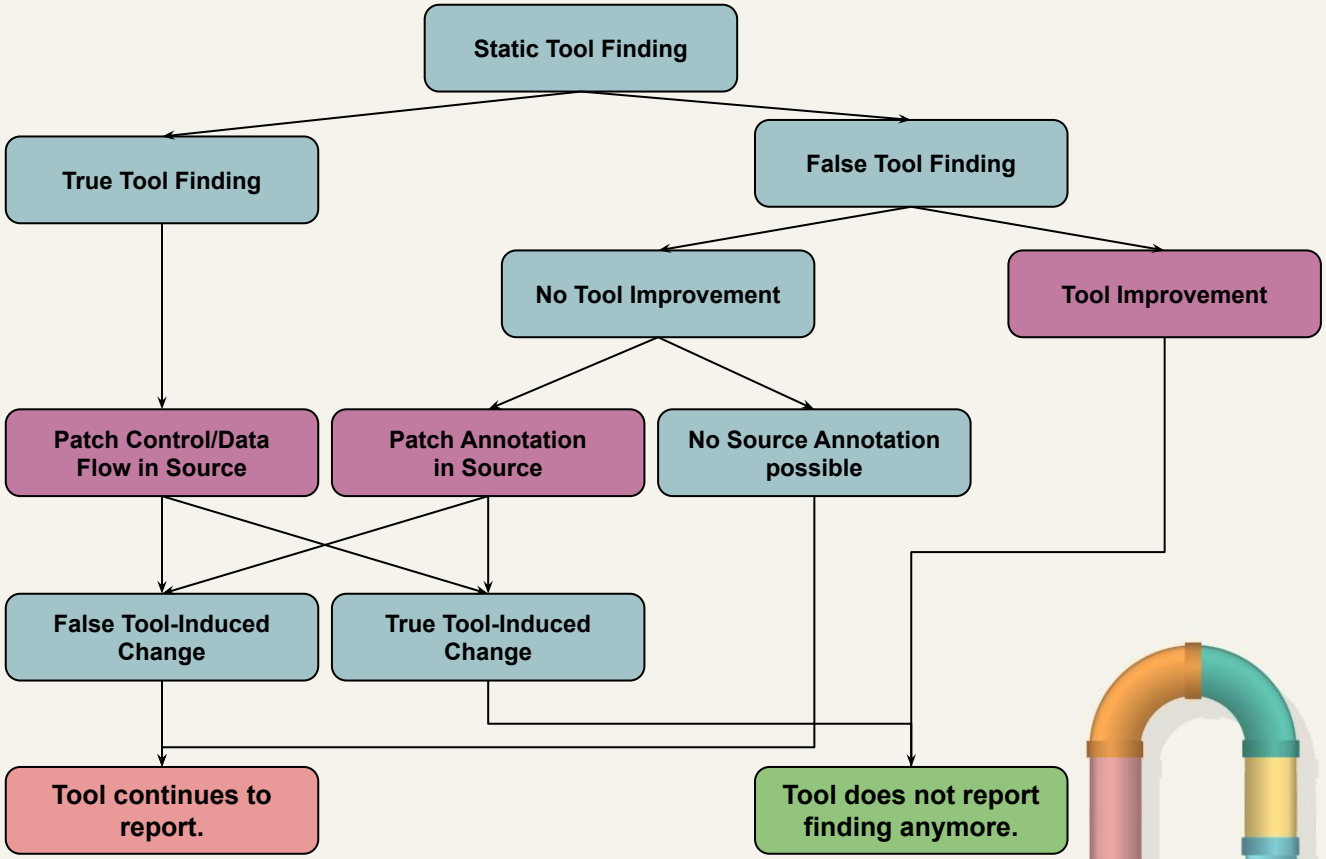


LINUX PLUMBERS CONFERENCE

August 24-28, 2020

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

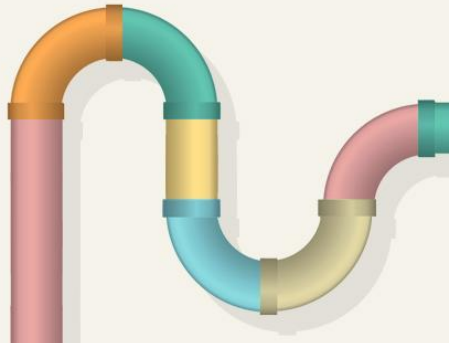
Despite all reasonable effort, there are cases where a static tool continues to report a finding and we must accept that we do not change the source or the tool.

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.



LINUX PLUMBERS CONFERENCE

August 24-28, 2020

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?

LINUX PLUMBERS CONFERENCE

August 24-28, 2020

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.

LINUX PLUMBERS CONFERENCE

August 24-28, 2020

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.

LINUX PLUMBERS CONFERENCE

August 24-28, 2020

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?

LINUX PLUMBERS CONFERENCE

August 24-28, 2020

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?

LINUX PLUMBERS CONFERENCE

August 24-28, 2020

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

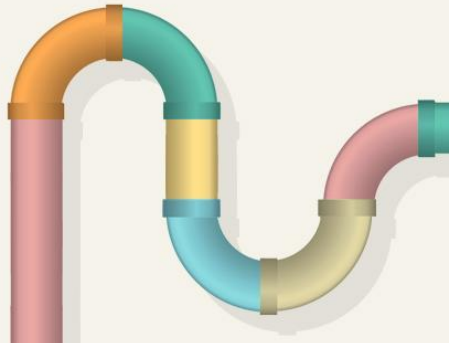
patch submission
enforceable.
partly done. little
infrastructure available.

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

kernel rc's
most observable
activity, IMHO.

kernel release
already introduces
churn of backporting
decision.

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.



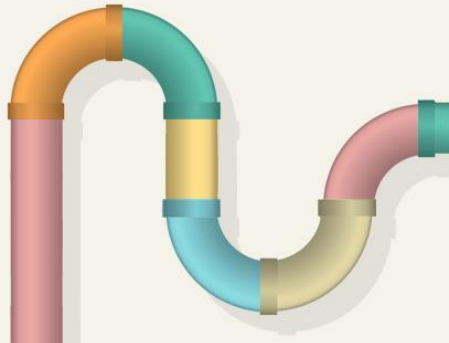
Technical Aspects and Solutions

Envisioned User Interface for the different phases:

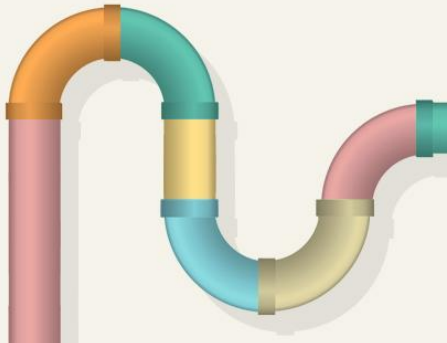
- Patch creation
- Discussion on mailing list
- While tracking various trees (linux-next, mainline, stable releases)

What are the preferred interfaces?

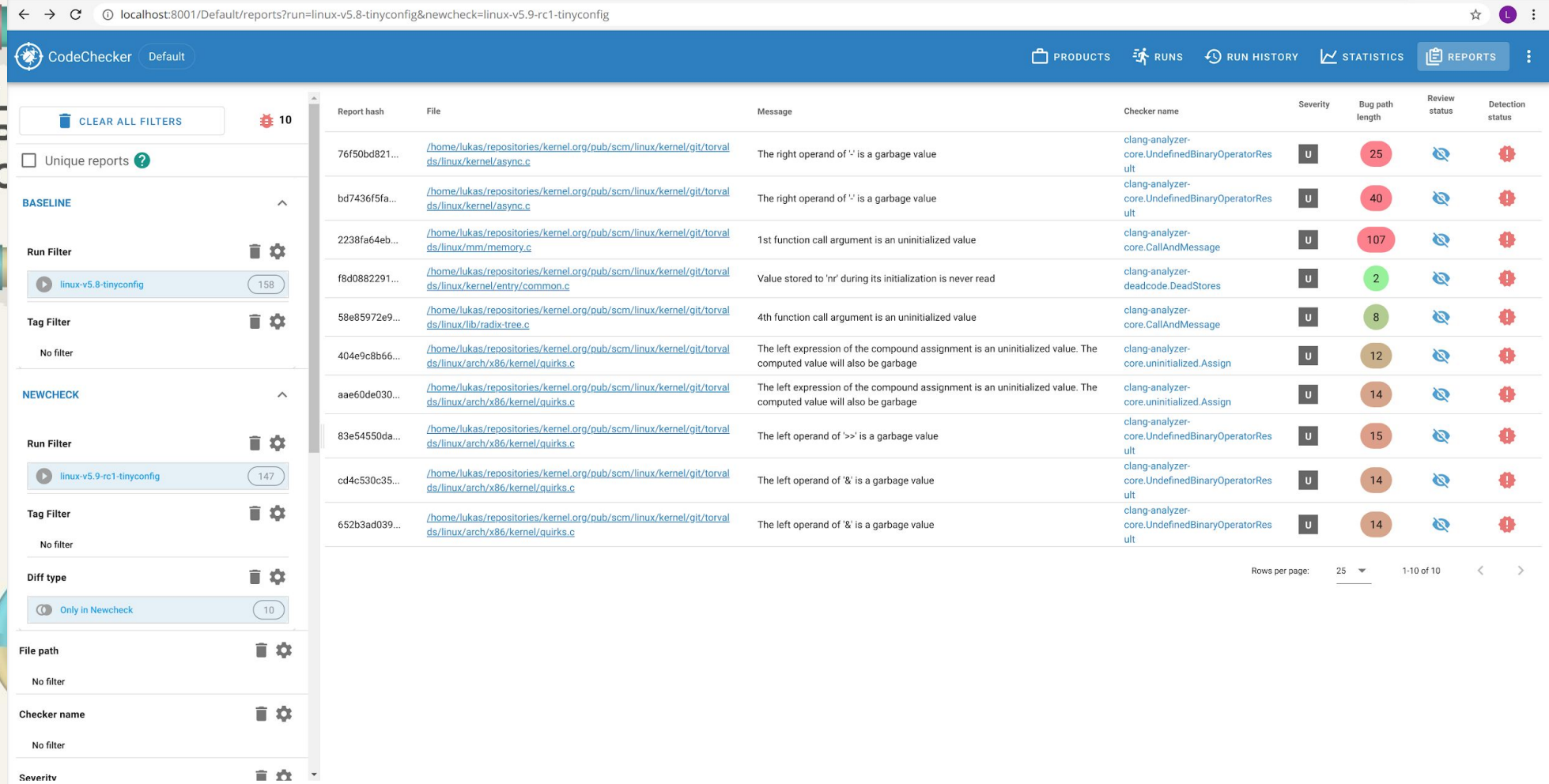
What are the core requirements for such a technical system?

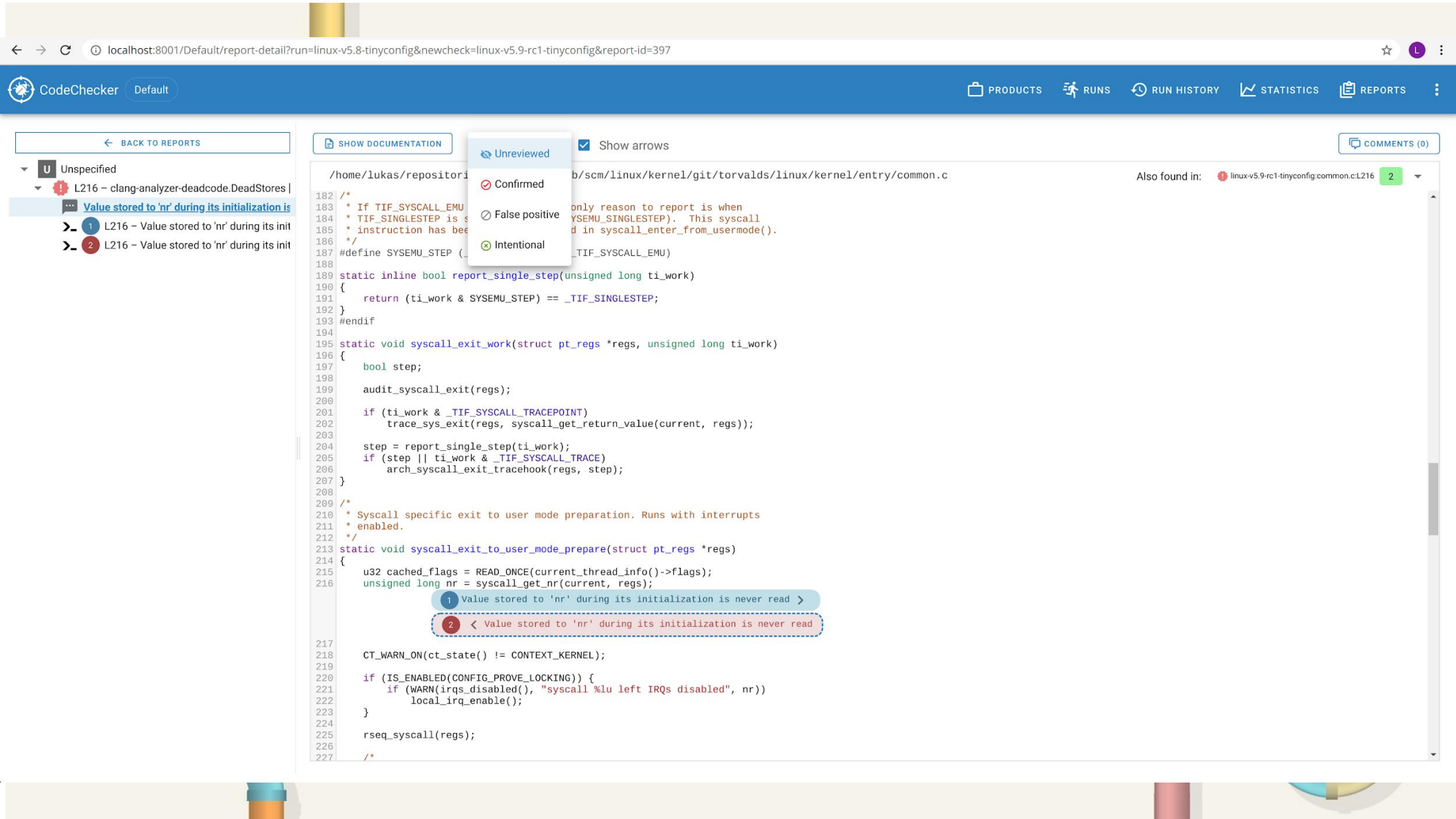


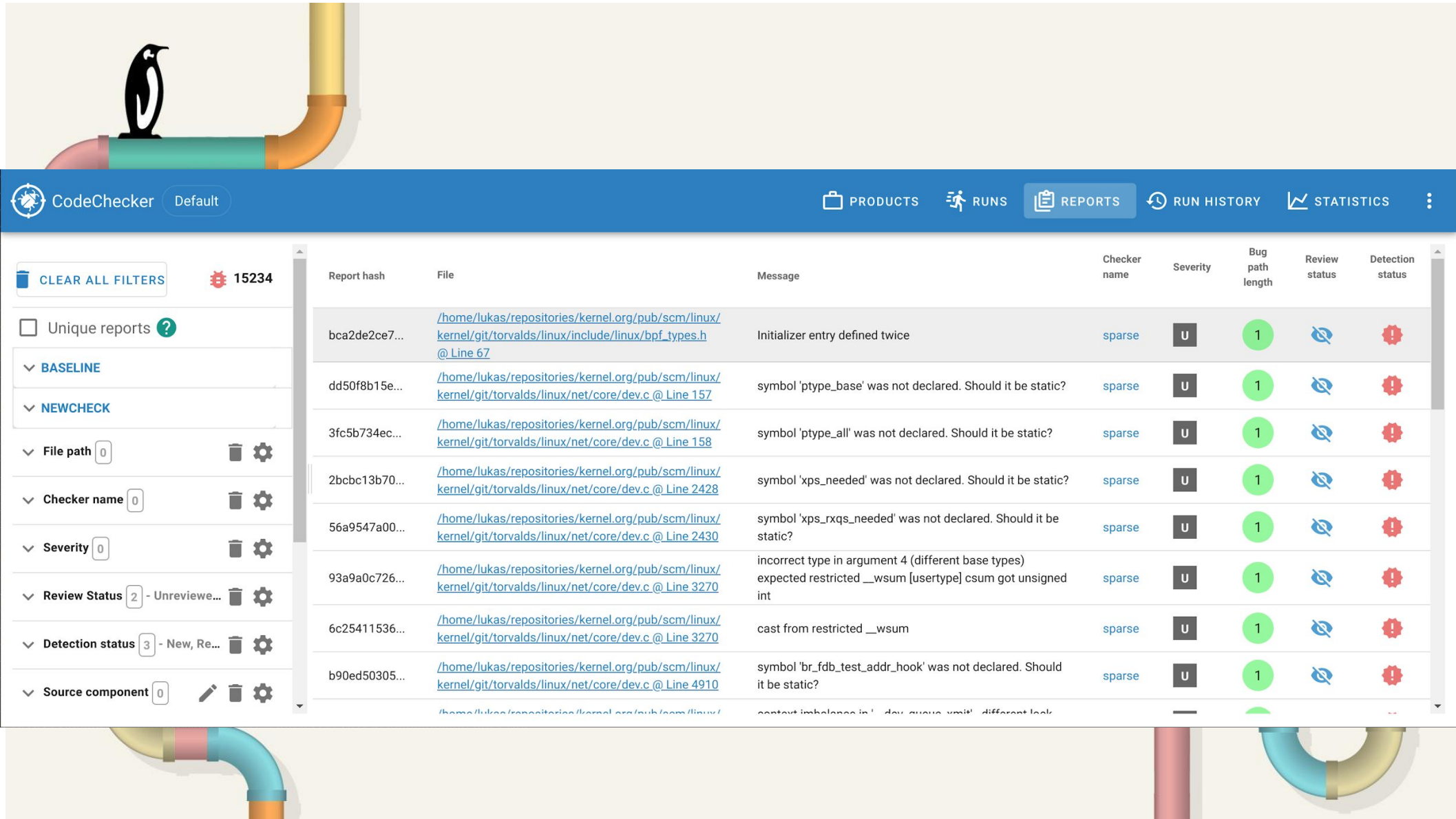
Backup



Some First Experiments with one Technical Solution




localhost:8001/Default/report-detail?run=linux-v5.8-tinyconfig&newcheck=linux-v5.9-rc1-tinyconfig&report-id=397
CodeChecker
Default
PRODUCTS
RUNS
RUN HISTORY
STATISTICS
REPORTS
BACK TO REPORTS
Unspecified
L216 – clang-analyzer-deadcode.DeadStores |
Value stored to 'nr' during its initialization is
L216 – Value stored to 'nr' during its init
L216 – Value stored to 'nr' during its init
SHOW DOCUMENTATION
Unreviewed
Confirmed
False positive
Intentional
Show arrows
COMMENTS (0)
/home/lukas/repositori
b/scm/linux/kernel/git/torvalds/linux/kernel/entry/common.c
Also found in:
linux-v5.9-rc1-tinyconfig:common.c:L216
2
182 /*
183 * If TIF_SYSCALL_EMU
only reason to report is when
184 * TIF_SINGLESTEP is s
YSEMU_SINGLESTEP). This syscall
185 * instruction has bee
d in syscall_enter_from_usermode().
186 */
187 #define SYSEMU_STEP (_
_TIF_SYSCALL_EMU)
188
189 static inline bool report_single_step(unsigned long ti_work)
190 {
191 return (ti_work & SYSEMU_STEP) == _TIF_SINGLESTEP;
192 }
193 #endif
194
195 static void syscall_exit_work(struct pt_regs *regs, unsigned long ti_work)
196 {
197 bool step;
198
199 audit_syscall_exit(regs);
200
201 if (ti_work & _TIF_SYSCALL_TRACEPOINT)
202 trace_sys_exit(regs, syscall_get_return_value(current, regs));
203
204 step = report_single_step(ti_work);
205 if (step || ti_work & _TIF_SYSCALL_TRACE)
206 arch_syscall_exit_tracehook(regs, step);
207 }
208
209 /*
210 * Syscall specific exit to user mode preparation. Runs with interrupts
211 * enabled.
212 */
213 static void syscall_exit_to_user_mode_prepare(struct pt_regs *regs)
214 {
215 u32 cached_flags = READ_ONCE(current_thread_info()->flags);
216 unsigned long nr = syscall_get_nr(current, regs);
1 Value stored to 'nr' during its initialization is never read
2 Value stored to 'nr' during its initialization is never read
217
218 CT_WARN_ON(ct_state() != CONTEXT_KERNEL);
219
220 if (IS_ENABLED(CONFIG_PROVE_LOCKING)) {
221 if (WARN(irqs_disabled(), "syscall %lu left IRQs disabled", nr))
222 local_irq_enable();
223 }
224
225 rseq_syscall(regs);
226
227 /*


CodeChecker
Default
PRODUCTS
RUNS
REPORTS
RUN HISTORY
STATISTICS
CLEAR ALL FILTERS
15234
Unique reports
BASELINE
NEWCHECK
File path 0
Checker name 0
Severity 0
Review Status 2 - Unreviewe...
Detection status 3 - New, Re...
Source component 0
Report hash
File
Message
Checker name
Severity
Bug path length
Review status
Detection status
bca2de2ce7...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/include/linux/bpf_types.h @ Line 67
Initializer entry defined twice
sparse
U
1
dd50f8b15e...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/net/core/dev.c @ Line 157
symbol 'ptype_base' was not declared. Should it be static?
sparse
U
1
3fc5b734ec...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/net/core/dev.c @ Line 158
symbol 'ptype_all' was not declared. Should it be static?
sparse
U
1
2bcbc13b70...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/net/core/dev.c @ Line 2428
symbol 'xps_needed' was not declared. Should it be static?
sparse
U
1
56a9547a00...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/net/core/dev.c @ Line 2430
symbol 'xps_rxqs_needed' was not declared. Should it be static?
sparse
U
1
93a9a0c726...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/net/core/dev.c @ Line 3270
incorrect type in argument 4 (different base types) expected restricted __wsum [usertype] csum got unsigned int
sparse
U
1
6c25411536...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/net/core/dev.c @ Line 3270
cast from restricted __wsum
sparse
U
1
b90ed50305...
/home/lukas/repositories/kernel.org/pub/scm/linux/kernel/git/torvalds/linux/net/core/dev.c @ Line 4910
symbol 'br_fdb_test_addr_hook' was not declared. Should it be static?
sparse
U
1