LF *live*

MENTORSHIP SERIES

LF Live MENTORSHIP SERIES

Smatch: New ideas for Static Analysis

Dan Carpenter, Oracle

LF live MENTORSHIP SERIES

Please download Smatch

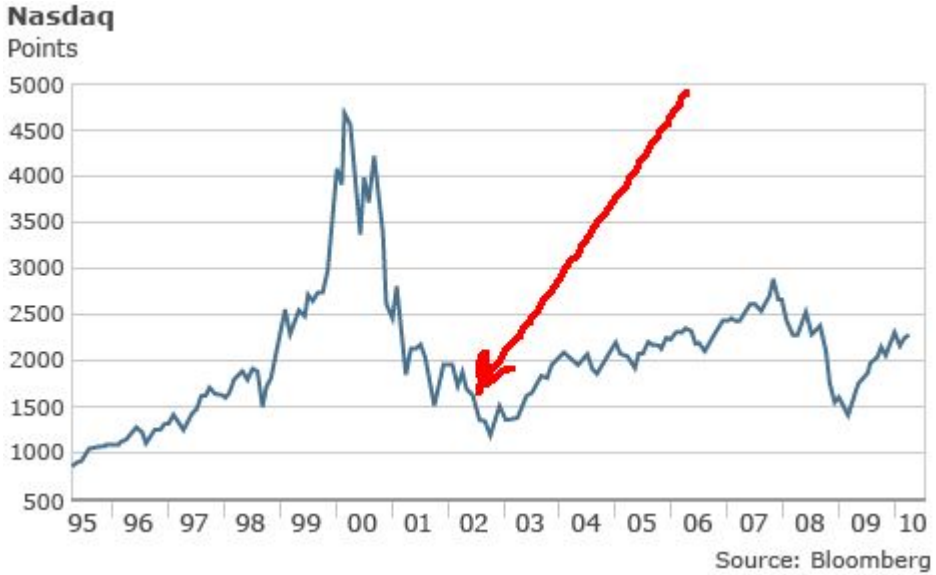
git clone https://github.com/error27/smatch.git

apt-get install gcc make sqlite3 libsqlite3-dev libdbd-sqlite3-perl libssl-dev libtry-tiny-perl

LF live MENTORSHIP SERIES

LF live MENTORSHIP SERIES

LF live MENTORSHIP SERIES



LF live MENTORSHIP SERIES

2010:

1 546 Author: Joe Perches <joe@perches.com>
2 510 Author: Dan Carpenter <error27@gmail.com> **<-- THIS IS ME!**
3 506 Author: Mauro Carvalho Chehab <mchehab@kernel.org>
4 493 Author: Chris Wilson <chris@chris-wilson.co.uk>
5 465 Author: Eric Dumazet <eric.dumazet@gmail.com>
6 412 Author: Paul Mundt <lethal@linux-sh.org>
7 395 Author: Mark Brown <broonie@opensource.wolfsonmicro.com>
8 383 Author: Johannes Berg <johannes.berg@intel.com>
9 381 Author: Greg Kroah-Hartman <gregkh@suse.de>
10 365 Author: Uwe Kleine-König <u.kleine-koenig@pengutronix.de>

2020:

1. 1230 Author: Mauro Carvalho Chehab <mchehab+huawei@kernel.org>
2. 1201 Author: Christoph Hellwig <hch@lst.de>
3. 1134 Author: Lee Jones <lee.jones@linaro.org>
4. 884 Author: Krzysztof Kozlowski <krzk@kernel.org>
5. 874 Author: Chris Wilson <chris@chris-wilson.co.uk>
6. 629 Author: Andy Shevchenko <andriy.shevchenko@linux.intel.com>
7. 523 Author: Colin Ian King <colin.king@canonical.com>
8. 521 Author: Randy Dunlap <rdunlap@infradead.org>
9. 507 Author: Sean Christopherson <seanjc@google.com>
10. 495 Author: Thomas Gleixner <tglx@linutronix.de>

Why Am I no longer a **Super Star** developer? :(

1. Laziness
2. Reporting more bugs instead of fixing them
3. Ran out of a backlog of bugs
4. Increased competition for easy bug fixes

2010:

1	78	Reported-by: Stephen Rothwell <sfr@canb.auug.org.au>
2	58	Reported-by: Ingo Molnar <mingo@elte.hu>
3	46	Reported-by: Randy Dunlap <randy.dunlap@oracle.com>
4	33	Reported-by: Dan Carpenter <error27@gmail.com> **<-- THIS IS ME!**
5	28	Reported-by: Linus Torvalds <torvalds@linux-foundation.org>
6	20	Reported-by: Andrew Morton <akpm@linux-foundation.org>
7	15	Reported-by: Stephane Eranian <eranian@google.com>
8	14	Reported-by: Johannes Berg <johannes@sipsolutions.net>
9	13	Reported-by: Jiri Slaby <jirislaby@gmail.com>
10	11	Reported-by: Jonathan Cameron <jic23@cam.ac.uk>

LF Live MENTORSHIP SERIES

2020:

1	769	Reported-by: Hulk Robot <hulkci@huawei.com>
2	507	Reported-by: kernel test robot <lkp@intel.com>
3	294	Reported-by: kbuild test robot <lkp@intel.com>
4	155	Reported-by: Dan Carpenter <dan.carpenter@oracle.com> ← **HERE I AM!**
5	126	Reported-by: Stephen Rothwell <sfr@canb.auug.org.au>
6	122	Reported-by: Randy Dunlap <rdunlap@infradead.org>
7	55	Reported-by: syzbot <syzkaller@googlegroups.com>
8	49	Reported-by: Naresh Kamboju <naresh.kamboju@linaro.org>
9	43	Reported-by: Qian Cai <cai@lca.pw>
10	38	Reported-by: kernel test robot <rong.a.chen@intel.com>

Main Static Analysis Tools Used in the Kernel

1. Sparse
    a. High quality code
    b. Fast
    c. Checks for endianness and user annotations
    d. Smatch uses Sparse as a C front end

Main Static Analysis Tools Used in the Kernel

1. Coccinelle
    a. Easy to write checks
    b. Works on un-preprocessed code (useful for checking macros)
    c. Generates patches automatically

Main Static Analysis Tools Used in the Kernel

1. Smatch
    a. Best flow analysis (in the world[1])
    b. Tracks the values of all the variables
    c. Cross function analysis
    d. Poorly documented
    e. Only works on the Linux kernel

[1] Important: Being the “best” is not the same as being “good”

Flow analysis: Math for how to understand code

1) Is this pointer NULL?
2) Is the lock held?
3) Is this pointer freed?

Impossible questions:

1) Bug or a feature?
2) Requires hardware knowledge?
3) Code requires too much resources to parse.

Can we determine the intention from reading the code?

```
if (x < 42) {  // ← Probably this means that x can be up to 0-41

if (!pointer) { // ← This means the author thinks the pointer can be NULL

pr_err("something");  // ← This means we have had an error
```

```
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  // ← OOPS! COPY AND PASTED FROM LINE BEFORE
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

warn: ignoring unreachable code.

warn: curly braces intended?

warn: inconsistent indenting

Sparse basics

1. Symbols (variables)
    a. struct symbol *sym;
    b. int a;
2. Expressions (varables, calls, assignments, addition, subtraction)
    a. struct expression *expr;
    b. a = 100
3. Statements
    a. struct statement *stmt;

Smatch basics

1. State
   a. struct smatch_state *state;
   b. STATE(freed);
2. Sm state (variable + state + pointers to history)
   a. struct sm_state *sm;
   b. set_state_expr(my_id, expr, &freed);
   c. state = get_state_expr(my_id, expr);
3. All the states are saved in a state tree
   a. struct stree *stree;

More Smatch basics

1. Numbers (type and value)
   a. sval_t sval;
   b. get_value(expr, &sval)
2. Range list (list of numbers) “1-4,6-9”
   a. struct range_list *rl;
   b. get_implied_rl(expr, &rl)
   c. get_absolute_rl(expr, &rl)
   d. get_user_rl(expr, &rl)

```
STATE(freed);

This check will automatically use the global states
&undefined and &merged
```

```
void check_free(int id)
{
    my_id = id;
    add_function_hook("kfree", &match_free, NULL);
    add_modification_hook(my_id, &modified);
    add_hook(&match_dereferences, DEREF_HOOK);
}
```

From the match_free() function:

```
arg = get_argument_from_call_expr(expr->args, 0);
set_state_expr(my_id, arg, &freed);
```

From the match_dereferences() function:

```
struct sm_state *sm;

sm = get_sm_state_expr(my_id, expr);

if (sm && slist_has_state(sm->possible, &freed))

        sm_warning("use after free");
```

Ideas 1: Use Smatch on something different

1. Run Smatch on your favourite C project
2. Run Smatch on all of Debian C code
3. Integrate Smatch into Jenkins or one of the other CI tools
4. Run Smatch on code from github
5. Create a twitter bot to complain about off by one array overflows

Ideas 2: Use Smatch to create something new

1. Create a website that gives people access to the cross function database
2. Integrate Smatch with a code editor
3. Use Smatch to generate test cases for Syzkaller
4. Take the ideas from Smatch like the cross function database and the flow analysis and port it to Clang/LLVM.

Ideas 3: Create a new check

1. Copy a check from Coccinelle
2. Sign up for a kernel mailing list and change the review comments into Smatch scripts
3. Add a --pedantic option to complain about style issues
4. `git log --no-merges -p v5.10`
5. Review last years CVEs (Use the ubuntu-cve-tracker)

Ideas 4: More ideas for checks

1. Create a “scheduling in atomic” warning. preempt_enable/disable()
2. Print a warning when we take user data and use it as an enum
3. Someone allocates memory with kmalloc() but frees it with devm_kfree()
4. Code checking IS_ERR() or IS_ERR_VALUE() but we already know the result
5. Other duplicative conditions especially if they lead to dead code

Ideas 5: More ideas for checks (very specific)

1. When a statement ends in a coma instead of a semi-colon and the following line is indented one less tab.

```
frob();
if (x == y)
    frob(),  // ← This comma is a bug
frob();
```

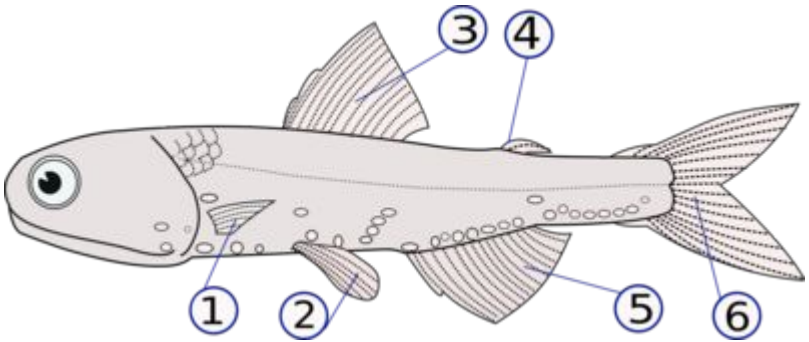
Ideas 6: Ideas that I should have given you but I instead implemented myself today

1) Propagating the error code from sscanf().
2) Returning success when sscanf gave us something unexpected
3) A buffer overflow from sscanf

```
ret = sscanf(buf, “%s %s”, first_part, last_part);
if (ret != 2)
    goto fail;
```

LF Live MENTORSHIP SERIES

Fin!



LF live MENTORSHIP SERIES

Thank you for joining us today!

We hope it will be helpful in your journey to learning more about effective and productive participation in open source projects. We will leave you with a few additional resources for your continued learning:

- The LF Mentoring Program is designed to help new developers with necessary skills and resources to experiment, learn and contribute effectively to open source communities.
- Outreachy remote internships program supports diversity in open source and free software
- Linux Foundation Training offers a wide range of free courses, webinars, tutorials and publications to help you explore the open source technology landscape.
- Linux Foundation Events also provide educational content across a range of skill levels and topics, as well as the chance to meet others in the community, to collaborate, exchange ideas, expand job opportunities and more. You can find all events at events.linuxfoundation.org.

LF live MENTORSHIP SERIES

Content Template 1

1. Abc
    a. Abc
    b. Abc
2. Abc
3. And so on