Finding and Fixing Bugs in Systems Code using Coccinelle

Julia Lawall (University of Copenhagen)
Gilles Muller (INRIA), René Rydhof Hansen (Aalborg),
Nicolas Palix (Grenoble)

November 11, 2011

Properties of large, legacy infrastructure software

- Low-level code
- Variable code quality
- May evolve rapidly
- May support many configurations
- Example: The Linux kernel

Problems of large, legacy infrastructure software

Bugs and defects:

- NULL pointer dereferences.
- Memory leaks, double free, use after free.
- Invalid lock management.
- Unreachable code.

Collateral evolutions:

- Changes in client code entailed by changes in API interfaces.
- Change of function name.
- Construction of new function arguments.
- Changes in the required order of operations.

Bug: dereference of a possibly NULL value

```
Author: Mariusz Kozlowski <m.kozlowski@tuxland.pl>

    tun/tap: Fix crashes if open() /dev/net/tun and
    then poll() it.

diff --git a/drivers/net/tun.c b/drivers/net/tun.c
@@ -486,12 +486,14 @@
- struct sock *sk = tun->sk;
+ struct sock *sk;
  unsigned int mask = 0;

  if (!tun)
    return POLLERR;

+ sk = tun->sk;
```

Collateral evolution: Refactoring of an API interface

```
Author: Jan Blunck <jblunck@suse.de>

    d_path: Make seq_path() use a struct path argument

    seq_path() is always called with a dentry and a vfsmount from a
    struct path.  Make seq_path() take it directly as an argument.

diff --git a/drivers/md/md.c b/drivers/md/md.c
@@ -5197,8 +5197,7 @@
-    seq_path(seq, bitmap->file->f_path.mnt,
-             bitmap->file->f_path.dentry," \t\n");
+    seq_path(seq, &bitmap->file->f_path, " \t\n");

diff --git a/mm/swapfile.c b/mm/swapfile.c
@@ -1394,7 +1394,7 @@
-  len = seq_path(swap, file->f_path.mnt, file->f_path.dentry, " \t\n\\");
+  len = seq_path(swap, &file->f_path, " \t\n\\");
```

Our goals

- Automatically find code containing bugs or defects, or requiring collateral evolutions.
- Automatically fix bugs or defects, and perform collateral evolutions.
- Provide a system that is accessible to software developers.

Requirements for automation

The ability to abstract over irrelevant information:

- Bug case: `struct sock *sk = tun->sk;`
- CE case: `bitmap->file->f_path.mnt` vs `file->f_path.mnt`

The ability to match scattered code fragments:

- Bug case: `struct sock *sk = tun->sk;` is a defect if followed by a NULL test on `tun`.

The ability to transform code fragments:

- CE case: Replace function arguments `X->Y.mnt` and `X->Y.dentry` by `&X->Y`.

Coccinelle

Program matching and transformation for unpreprocessed C code.

Fits with the existing habits of C programmers.

- C-like, patch-like notation

Semantic patch language (SmPL):

- Metavariables for abstracting over subterms.
- "..." for abstracting over code sequences.
- Patch-like notation ($-$/$+$) for expressing transformations.

Bug finding (and fixing)

```
@@
type T;
identifier i,fld;
expression E;
statement S;
@@

  T i = E->fld;

  ...

  if (E == NULL) S
```

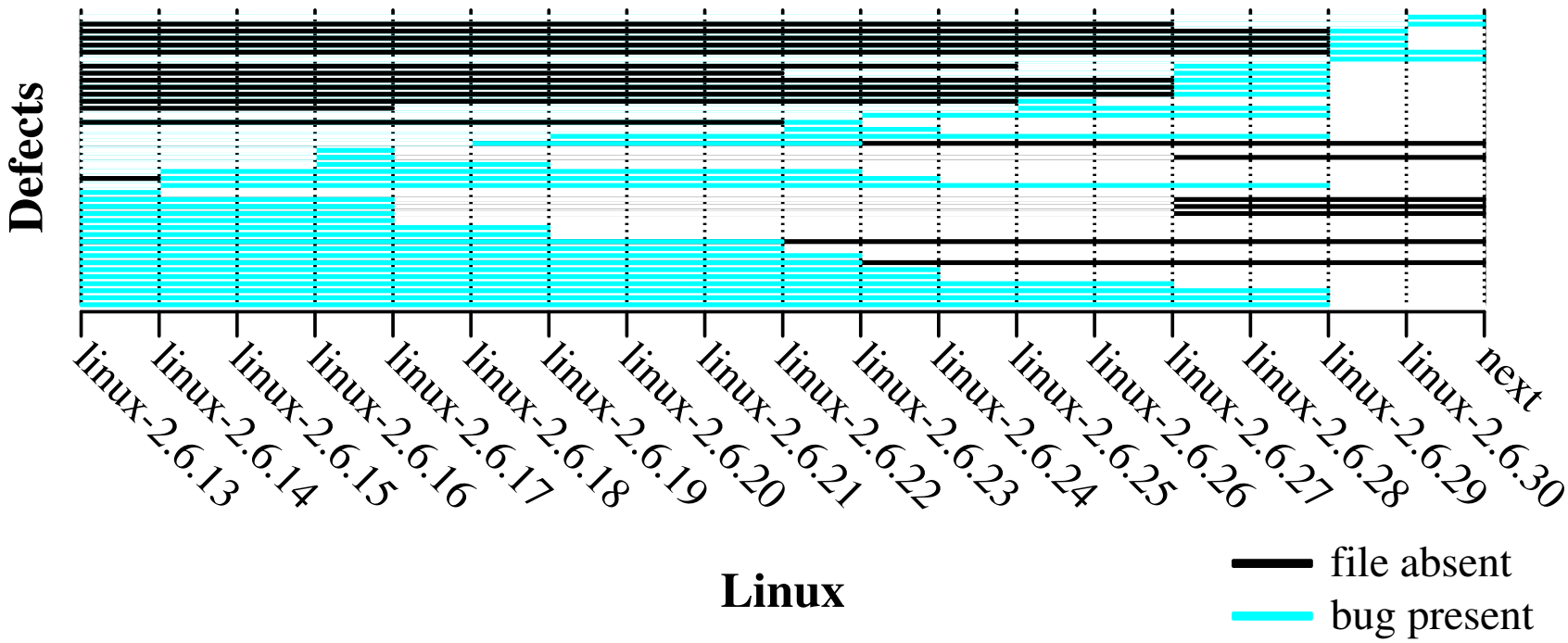
Bug finding (and fixing)

```
@@
type T;
identifier i,fld;
expression E;
statement S;
@@

  T i = E->fld;

  ... when != E
      when != i
  if (E == NULL) S
```

Bug finding (and fixing)

```
@@
type T;
identifier i,fld;
expression E;
statement S;
@@

- T i = E->fld;
+ T i;
  ... when != E
      when != i
  if (E == NULL) S
+ i = E->fld;
```

Potential impact of the semantic patch



Collateral evolution

```
@@
expression file,E1,E2;
identifier fld;
@@

 seq_path(E1,
-             file->fld.mnt,file->fld.dentry
+             &file->fld
          ,E2)
```

Updates 11/12 relevant code sites.

This rule can be automatically generated from examples.

Current status

- Over 750 patches based on Coccinelle accepted into the Linux kernel.
- A collection of semantic patches integrated into the Linux kernel source tree.
- Several LWN articles by Linux developers.
- Seems to be "easy" to learn.
- Used by developers of Linux and other software.
- Articles in EuroSys, DSN, POPL, ASE, AOSD, etc. on the language and methodology.

Conclusion

- Coccinelle provides a declarative language for program matching and transformation.
- Coccinelle semantic patches look like patches; fit with Linux programmers' habits.
- Quite "easy" to learn; already accepted by the Linux community.
- Future work will build on Coccinelle to develop tools motivated by problems observed in Linux development.

http://coccinelle.lip6.fr