

Effective Static Analysis Shift Left for the Development of Safe and Secure Embedded Systems

Roberto Bagnara, Abramo Bagnara

bugSeng

<http://bugseng.com>

& Applied Formal Methods Laboratory
Department of Mathematical, Physical and Computer Sciences
University of Parma, Italy

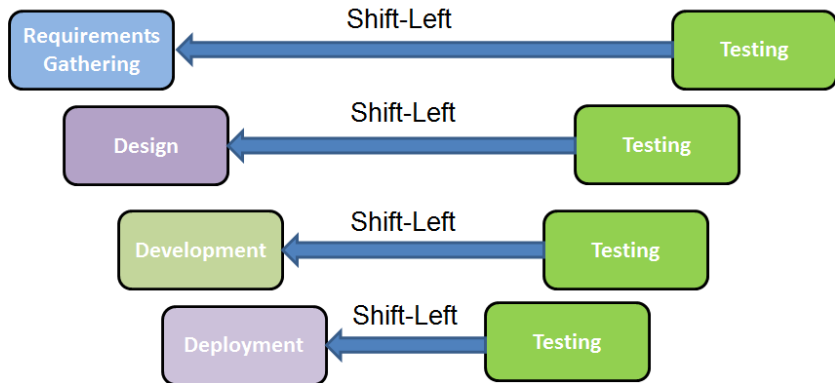
IWES 2022

Bari, September 23rd, 2022

Outline I

- 1 Shift-left
- 2 Example: Enforcing the MISRA Guidelines
- 3 Features Coming from CI Systems
- 4 Features the SAST Tools Must Provide
- 5 Conclusion

Shift-left



Shift-left: Not a New Idea

Shift-left dates back to **2001**:

*“Shift-left testing is how I refer to a **better way of integrating the quality assurance (QA) and development** parts of a software project. By linking these two functions at lower levels of management, you can expand your testing program while **reducing manpower and equipment needs** — sometimes by as much as an order of magnitude.”*

— Larry Smith, Dr. Dobb's Journal, September 1st, 2001

Shift-left: Not Just for Classical Testing

Today the notion of *testing* has to be interpreted in a **broad sense**, including **all sorts of static and dynamic analysis techniques**

$$\text{SAST} \implies \left\{ \begin{array}{l} \text{Static Application Security Testing} \\ \text{Static Application Safety Testing} \end{array} \right.$$

Software issues are better discovered earlier rather than later:

"Bugs are cheap when caught young."

— Larry Smith, *ibidem*

Example: The MISRA C/C++ Coding Standards

The MISRA coding standards are the most authoritative sets of guidelines for the development of safe and secure systems in C/C++

The **highest payoff** from the adoption of the MISRA Guidelines is achieved when they are **adopted at the very beginning** of a project. . .

. . . and it is **systematically enforced** with the help of a **high-quality tool**

Imposing MISRA C on an existing code base with a proven track record **may be counterproductive** if not done properly. . .

. . . this requires **significant expertise** and **tools of even higher quality** (powerful deviation mechanisms, baselining, . . .)

Example: The MISRA C/C++ Coding Standards (cont'd)

Early adoption of the MISRA Guidelines is not greatly facilitated thanks to the use of Continuous Integration (CI) systems:

- Jenkins
- GitLab CI
- Bamboo
- ...

Basic functionality: whenever a change is committed into the source code repository, static analysis is triggered, and developers have access to the analysis results via the web

Important Features Coming from the CI System

- Keeping track of the last k analyses
- **Freezing** the results of particular analyses
- Showing **trends** (e.g., in the number of diagnostic messages)
- Triggering a new analysis (if not automatically done at each commit)
- **Quality gates** on the submitted patches, e.g.. developers commit to a “triage area” and this triggers execution of the static analysis tool
 - if the number of violations for the selected set does not increase, then the change is **automatically merged** into the development branch
 - otherwise an **email alert is sent** to the author of the change

Key Features of the Static Analysis Software

Among the advantages of deploying SAST via CI is the fact that users are **spared from configuration and execution** of the static analysis tool

Other features are crucial for successful adoption, and only the static analysis tool can provide them (or not):

- 1 Users must have access to **fully detailed reports** (e.g., Sarif as implemented, e.g., in GitHub is inadequate)
- 2 Each user must be able to use **private, sophisticated filters** (i.e., locally-stored and independent from one another)
- 3 Each user must be able to use **his/her favorite IDE** (Eclipse, Visual Studio, Visual Studio Code, NetBeans, CLion, ...)

Access To Fully-Detailed Reports from Anywhere

The screenshot shows a web browser window displaying a code editor with C code and a list of static analysis violations. The code includes variables like `oldstroffset`, `new`, `uint32_t tag`, and `offset`, and a function `FDT_SW_PROBE_STRUCT`. The violations are for MC3R1.R13.4, which is a non-compliant '=' assignment operator. The violations are detailed, showing the specific code lines and the context of the errors.

```

341     int oldstroffset, new;
342     uint32_t tag;
343     int offset, nextoffset;
344
345     FDT_SW_PROBE_STRUCT(fdt);
346
347     /* Add terminator */
348     end = fdt_grab_space(fdt, sizeof(*end));

```

Violations for MC3R1.R13.4 (non-compliant '=' assignment operator):

- lib/libfdt/fdt_sw.c:299.9-299.32: result is used as left operand of '!=' inequality operator
- <preprocessed lib/libfdt/fdt_sw.c>:597.51-597.52: preprocessed tokens
- lib/libfdt/fdt_sw.c:79.55-79.56: expanded from macro 'FDT_SW_PROBE_STRUCT'
- lib/libfdt/fdt_sw.c:345.9-345.32: non-compliant '=' assignment operator
- <preprocessed lib/libfdt/fdt_sw.c>:635.22: preprocessed tokens
- lib/libfdt/fdt_sw.c:79.26: expanded from macro 'FDT_SW_PROBE_STRUCT'
- lib/libfdt/fdt_sw.c:345.9-345.32: result is used as left operand of '!=' inequality operator
- <preprocessed lib/libfdt/fdt_sw.c>:635.51-635.52: preprocessed tokens
- lib/libfdt/fdt_sw.c:79.55-79.56: expanded from macro 'FDT_SW_PROBE_STRUCT'
- lib/libfdt/fdt_sw.c:361.21: non-compliant '=' assignment operator
- lib/libfdt/fdt_sw.c:361.63-361.64: result is used as left operand of '!=' inequality operator

Access To Fully-Detailed Reports from Anywhere (cont'd)

631 fdt32_t *end; `<preprocessed lib/libfdt/fdt_sw.c>`
 632 int oldstroffset, newstroffset;
 633 uint32_t tag;
 634 int offset, nextoffset;
 635 { int err; if ((err = fdt_sw_probe_struct(fdt)) != 0) return err; };

≡ MC3R1.R13.4 preprocessed tokens

636 end = fdt_grab_space(fdt, sizeof(*end));
 637 if (! end)
 638 return -3;

[lib/libfdt/fdt_sw.c:299.9-299.32:](#) result is used as left operand of `!` inequality operator
[<preprocessed lib/libfdt/fdt_sw.c>:597.51-597.52:](#) preprocessed tokens
[lib/libfdt/fdt_sw.c:79.55-79.56:](#) expanded from macro `FDT_SW_PROBE_STRUCT`

≡ violation for MC3R1.R13.4 untagged

[lib/libfdt/fdt_sw.c:345.9-345.32:](#) non-compliant `=` assignment operator
[<preprocessed lib/libfdt/fdt_sw.c>:635.22:](#) preprocessed tokens
[lib/libfdt/fdt_sw.c:79.26:](#) expanded from macro `FDT_SW_PROBE_STRUCT`
[lib/libfdt/fdt_sw.c:345.9-345.32:](#) result is used as left operand of `!` inequality operator
[<preprocessed lib/libfdt/fdt_sw.c>:635.51-635.52:](#) preprocessed tokens
[lib/libfdt/fdt_sw.c:79.55-79.56:](#) expanded from macro `FDT_SW_PROBE_STRUCT`

≡ violation for MC3R1.R13.4 untagged

[lib/libfdt/fdt_sw.c:361.21:](#) non-compliant `=` assignment operator
[lib/libfdt/fdt_sw.c:361.63-361.64:](#) result is used as left operand of `!` inequality operator

https://eclairit.com:3787/!s/var/lib/jenkins/jobs/TF-A/configurations/axis-CONF/fvp_ENABLE_RME/axis-agent/public/builds/133/archive/ECLAIR/out/PROJECT.ecd/fv_service/MC3R1.R13.4.html#R5057_2

Individual Filtering Capabilities

Section 6.7: "An empty declaration." [STD.emptdecl]). Tool used is ``/opt/gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'`

violation for MC3R1.R1.1 untagged
[include/common/bl_common.h:134:57:](#) empty declaration (ill-formed for the C99 standard, ISO/IEC Section 6.7: "An empty declaration." [STD.emptdecl]). Tool used is ``/opt/gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'`

violation for MC3R1.R1.1 untagged
[include/common/bl_common.h:153:70:](#) empty declaration (ill-formed for the C99 standard, ISO/IEC Section 6.7: "An empty declaration." [STD.emptdecl]). Tool used is ``/opt/gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'`

violation for MC3R1.R1.1 untagged
[include/common/bl_common.h:154:69:](#) empty declaration (ill-formed for the C99 standard, ISO/IEC Section 6.7: "An empty declaration." [STD.emptdecl]). Tool used is ``/opt/gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'`

violation for MC3R1.R1.1 untagged
[include/common/debug.h:35:10-35:18:](#) `#include <stdio.h>` directive is used in a freestanding environment for the C99 standard, ISO/IEC 9899:1999 Annex J.3.12 item 1: "Any library facilities available to freestanding program, other than the minimal set required by clause 4 (5.1.2.1)." [STD.freestlb]). Tool used is ``/opt/gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'`

violation for MC3R1.R1.1 untagged
[include/common/debug.h:58:35:](#) token pasting of `'` and `__VA_ARGS__` is a GNU extension (ill-formed for the C99 standard, ISO/IEC 9899:1999: "An ill-formed source detected by the parser." [STD.diag/ext_paste]). Tool used is ``/opt/gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'`

Individual Filtering Capabilities (cont'd)

The screenshot shows a web browser displaying a list of compiler warnings. Three warnings are highlighted with red boxes. The first two warnings are identical, and the third is different. A configuration menu for ECLAIR is overlaid on the right side of the page.

Warning 1: violation for MC3R1.R1.1 untagged
[lib/gpt_rme/gpt_rme.c:1235.25-1235.34:](#) binary integer literal
 9899:1999 Section 6.4.4: "An integer literal with prefix `0b'."
 2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'
[<preprocessed lib/gpt_rme/gpt_rme.c>:1522.19-1522.23:](#)
[<scratch space>:356.1-356.5:](#) expanded from macro '[include/export/lib/utls_def_exp.h:28.26-28.30:](#)
[include/export/lib/utls_def_exp.h:29.25-29.30:](#)
[include/lib/gpt_rme/gpt_rme.h:40.41-40.47:](#) expanded from macro '[include/export/lib/utls_def_exp.h:28.26-28.30:](#)

Warning 2: violation for MC3R1.R1.1 untagged
[lib/gpt_rme/gpt_rme.c:1235.25-1235.34:](#) binary integer literal
 9899:1999 Section 6.4.4: "An integer literal with prefix `0b'."
 2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'
[<preprocessed lib/gpt_rme/gpt_rme.c>:1535.19-1535.23:](#)
[<scratch space>:356.1-356.5:](#) expanded from macro '[include/export/lib/utls_def_exp.h:28.26-28.30:](#)
[include/export/lib/utls_def_exp.h:29.25-29.30:](#)
[include/lib/gpt_rme/gpt_rme.h:40.41-40.47:](#) expanded from macro '[include/export/lib/utls_def_exp.h:28.26-28.30:](#)

Warning 3: violation for MC3R1.R1.1 untagged
[lib/libc/abort.c:7.10-7.19:](#) `#include <stdlib.h>` directive is
 C99 standard, ISO/IEC 9899:1999 Annex J.3.12 item 1: "Any
 other than the minimal set required by clause 4 (5.1.2.1)." [STD.freestlib]). Tool used is `/opt/gcc-arm-10`
 2020.11-x86_64-aarch64-none-elf/bin/aarch64-none-elf-gcc'

ECLAIR Configuration Menu:

- Options
- Preview window Mouse over preview
- Current selection [Abramo](#)
- Area kinds
 - culprits
 - evidence
- Subarea kinds
 - expanded
 - preprocessed
- Hide reports where
 - any of the following are true:
 - `^.*__builtin_offsetof`
 - `^.*empty declaration`
 - message is one of:

Use of Favorite IDEs

Users must be able to use their favorite IDE to act upon the analysis findings, which entails:

- 1 The IDE plugin must be able to relate the local source code and the remotely analyzed source code
- 2 The IDE plugin must take into account line insertions and removal to adapt the location information of the involved code areas

Use of Favorite IDEs (cont'd)

The screenshot shows an IDE window with the ECLAIR extension. The left sidebar contains a menu with 'ECLAIR REPORT' expanded, showing a violation for rule MC3R1.R10.3. A tooltip is displayed over the code, providing details about the violation: 'common/fdt_wrappers.c: call to function 'fdt32_to_cpu(fdt32_t)' (unit 'common/fdt_wrappers.c' is 'assignment operator expected' and 'call to function 'fdt32_to_cpu(fdt32_t)' (unit 'common/fdt_wrappers.c' with target 'build/fvp/release/bl1/fdt_wrappers.o) has essential type 32-bit unsigned integer and standard type 'uint32_t' (that is 'unsigned)'. The code in the background includes C code for bus node translation.

```

File Edit Selection View Go Run Terminal Help
ECLAIR ... Extension: ECLAIR Settings @fdt_wrappers.c X
ECLAIR COMMANDS
  Disable 443
  Stop server 444
  Pair Browser 445
  First report 446
  Previous report 447
  Next report 448
  Last report 449
  Previous area 450
  Next area 451
  Add tagging comment 452
  Run 453
  Get on browser 454
ECLAIR REPORT
  violation for rule MC3R1.R10.3 (The value of an expression shall not ... 455
  common/fdt_wrappers.c call to function 'fdt32_to_cpu(fdt32_t)' (unit 'co... 456
  common/fdt_wrappers.c 'is' assignment operator expected 457
  common/fdt_wrappers.c: call to function 'fdt32_to_cpu(fdt32_t)' (unit 458
  'common/fdt_wrappers.c' with target 'build/fvp/release/bl1/fdt_wrappers.o) has 459
  essential type 32-bit unsigned integer and standard type 'uint32_t' (that is 460
  'unsigned') 461
  ranges_prop->data; 462
  the "ranges" +/ 463
  for (int i = 0; i < ncells_xlat; i++) { 464
    if (fdtw_xlat_hit(next_entry, self_addr_cells, 465
      parent_addr_cells, self_size_cells, 466
      &translated_addr)){ 467
      return translated_addr; 468
    } 469
    next_entry = next_entry + ncells_xlat; 470
  } 471
  } 472
  } 473
Ln 458, Col 48 (30 selected) Tab Size: B C ECLAIR

```

Conclusion

Shift-left adoption of static analysis tools is highly beneficial

- it reduces reworking
- it decreases time to completion

If the static analysis tool supports **high-quality output delivery over the network**, much can be done by the CI system itself

But for some key features the static analysis tool must have been designed **carefully and with remote deployment in mind**

ECLAIR has recently been chosen as the solution for static analysis in Continuous Integration systems of **Trusted Firmware**

Visit eclairit.com and make up your mind on the subject

Download a trial version of the **ECLAIR Client Kit** to play with the IDEs integrations



On eclairit.com you can see what it looks like to use ECLAIR on an integration server and browse the reports directly in your browser, without having to install ECLAIR on your PC. No registration is required. Several popular open-source projects are here analyzed for compliance with MISRA and other coding standards. Among those: Linux-next, bitcoin, NASA Core Flight System, ARM Mbed OS, ARM Trusted Firmware A, XEN hypervisor, Zephyr RTOS.

JENKINS

[SEE ECLAIR IN ACTION WITH JENKINS](#)

GITLAB

[SEE ECLAIR IN ACTION WITH GITLAB](#)

ECLAIR CLIENT KIT

Want to use this service from within your favorite IDE?

[TRY ECLAIR CLIENT KIT](#)

The End



Questions?

roberto.bagnara@unipr.it

info@bugsend.com

bugS**eng**