



## Static Analysis with the Yocto Project

Jan-Simon Möller, [dl9pf@gmx.de](mailto:dl9pf@gmx.de)

# Intro

Dipl.-Ing.  
Jan-Simon Möller

dl9pf on freenode  
[dl9pf@gmx.de](mailto:dl9pf@gmx.de)

AGL Release Manager  
[jsmoeller@linuxfoundation.org](mailto:jsmoeller@linuxfoundation.org)

OpenEmbedded & Yocto Project Board Member



# Topics

- Static Analysis - whaaaat ?
- Overview of tools
- CodeChecker
- meta-codechecker
- Summary
- Q/A



**Static Analysis - whaaaat ?**

**& why you should use it !**

# Static Analysis - whaaaat ?

- Static Analysis is a method to analyse a program that is performed without actually executing programs.
- Static Analysis becomes an increasingly important topic when the project involves Functional Safety aspects. This is the case in Automotive and in Automation as well.
- "But of course /MY code is always correct. "
  - But the auditor needs a way to (ap)prove that!

# Motivation

- Static analysis will not solve all problems (™).
- It will help catching some (possibly tricky to find) bugs.
- The goal is to show ways how to do this using open source tools available.
- Possible integrations are presented
- I will introduce basics but focus on what can be integrated with  
OpenEmbedded / The Yocto Project builds.



# Overview of static analysis tools

# Overview of tools

- There are tools available as OSS and proprietary tools.
- Some do pattern recognition, some use/enhance compilers, some are simple scripts. OSS tools include:
- gcc
- clang
- cppcheck
- flawfinder
- rats
- split



# During development you can easily use these directly within your source tree:

- **gcc (since gcc 10)**
  - gcc -fanalyzer
- **clang**
  - e.g. scan-build make
- **cppcheck**

gcc -fanalyzer enables:

- Wanalyzer-double-fclose
- Wanalyzer-double-free
- Wanalyzer-exposure-through-output-file
- Wanalyzer-file-leak
- Wanalyzer-free-of-non-heap
- Wanalyzer-malloc-leak
- Wanalyzer-possible-null-argument
- Wanalyzer-possible-null-dereference
- Wanalyzer-null-argument
- Wanalyzer-null-dereference
- Wanalyzer-stale-setjmp-buffer
- Wanalyzer-tainted-array-index
- Wanalyzer-unsafe-call-within-signal-handler
- Wanalyzer-use-after-free
- Wanalyzer-use-of-pointer-in-stale-stack-frame



```
> gcc -Werror -fanalyzer nullpointer.c
nullpointer.c: In function 'main':
nullpointer.c:7:5: error: dereference of NULL 'pointer' [CWE-690] [-Werror=analyzer-null-dereference]
    7 | int value = *pointer; /* Dereferencing happens here */
      |           ^~~~~
'main': events 1-2
|
|     6 | int * pointer = NULL;
|       |     ^~~~~~
|       |     |
|       |     (1) 'pointer' is NULL
|     7 | int value = *pointer; /* Dereferencing happens here */
|       |     ~~~~~
|       |     |
|       |     (2) dereference of NULL 'pointer'
|
cc1: all warnings being treated as errors
```

# clang (clang-tidy)

```
> clang-tidy nullpointer.c
```

```
Running without flags.
```

```
2 warnings generated.
```

```
nullpointer.c:7:5: warning: Value stored to 'value' during its initialization is never read  
[clang-analyzer-deadcode.DeadStores]
```

```
int value = *pointer; /* Dereferencing happens here */  
      ^
```

```
nullpointer.c:7:5: note: Value stored to 'value' during its initialization is never read
```

```
nullpointer.c:7:13: warning: Dereference of null pointer (loaded from variable 'pointer')  
[clang-analyzer-core.NullDereference]
```

```
int value = *pointer; /* Dereferencing happens here */  
      ^
```

```
nullpointer.c:6:1: note: 'pointer' initialized to a null pointer value
```

```
int * pointer = NULL;  
^
```

```
nullpointer.c:7:13: note: Dereference of null pointer (loaded from variable 'pointer')
```

```
int value = *pointer; /* Dereferencing happens here */  
      ^
```

# clang (scan-build)

```
> scan-build make
```

```
scan-build: Using '/usr/bin/clang-10.0.1' for static analysis
/usr/bin/clang-analyzer -c nullpointer.c -o nullpointer
```

```
nullpointer.c:7:5: warning: Value stored to 'value' during its initialization is never read
int value = *pointer; /* Dereferencing happens here */
    ^~~~~ ~~~~~~
```

```
nullpointer.c:7:13: warning: Dereference of null pointer (loaded from variable 'pointer')
int value = *pointer; /* Dereferencing happens here */
            ^~~~~~
```

```
2 warnings generated.
```

```
scan-build: 2 bugs found.
```

```
scan-build: Run 'scan-view /tmp/scan-build-2020-10-15-161857-10509-1' to examine bug reports.
```

```
> scan-view /tmp/scan-build-2020-10-15-161857-10509-1
Starting scan-view at: http://127.0.0.1:8181
```

(-> point browser to this)

```
1  #include <stddef.h>
2
3  int main(int argc, char *argv[]) {
4
5
6  int * pointer = NULL;
7
8  int value = *pointer; /* Dereferencing happens here */
9
10 return 0;
11 }
12
```

1 'pointer' initialized to a null pointer value -->

2 -- Dereference of null pointer (loaded from variable 'pointer')

# cppcheck

```
> cppcheck nullpointer.c
Checking nullpointer.c ...
nullpointer.c:7:14: error: Null pointer dereference: pointer
[nullPointer]
int value = *pointer; /* Dereferencing happens here */
            ^

nullpointer.c:6:17: note: Assignment 'pointer=NULL', assigned value is 0
int * pointer = NULL;
            ^

nullpointer.c:7:14: note: Null pointer dereference
int value = *pointer; /* Dereferencing happens here */
            ^
```



**meta-clang**

**Cool, I want that for my builds ...**

# Allright, let's talk about clang and meta-clang !!

meta-clang is a layer that adds support for the Clang Compiler/Toolchain.

TLDR: software can be compiled with clang instead of gcc

Thus we can enable tooling like scan-build right away.

More details: [https://elinux.org/images/3/3a/ELC\\_2020\\_clang.pdf](https://elinux.org/images/3/3a/ELC_2020_clang.pdf)

# meta-clang

URL: <https://github.com/kraj/meta-clang>

## Main features:

- adds clang as compiler - selectively or
- clang for everything\* (TOOLCHAIN = "clang")
- use compiler-rt as runtime (RUNTIME = "llvm")
- add clang to SDK (CLANGSDK = "1")
- enable scan-build (INHERIT += "scan-build")



# meta-clang

## Notes:

- Look at `meta-clang/conf/nonclangable.conf`
  - e.g. glibc, gcc, u-boot, grub
- `meta-clang/conf/nonscanable.conf`

## meta-clang in action

```
git clone git://git.yoctoproject.org/poky
```

```
git clone https://github.com/kraj/meta-clang
```

```
source poky/oe-init-build-env
```

```
bitbake-layers add-layer ../meta-clang
```

## meta-clang in action (continued)

```
cat << EOF >> conf/site.conf
```

```
TOOLCHAIN = "clang"
```

```
CLANGSDK = "1"
```

```
EOF
```

## meta-clang in action (continued)

```
bitbake core-image-minimal
```

option:

```
bitbake -c populage_sdk core-image-minimal
```

## meta-clang in action (continued)

To enable scan-build do:

```
cat << EOF >> conf/site.conf
```

```
INHERIT += "scan-build"
```

```
SCAN_BUILD = ""
```

```
SCAN_BUILD_pn-busybox = "1"
```

```
CLANG_SCAN_SERVER_IP ??= "0.0.0.0"
```

```
EOF
```

## meta-clang in action (continued)

To do a scan:

```
bitbake busybox
```

or

```
bitbake -c scanbuild busybox
```

## meta-clang in action (continued)

To view the results:

```
bitbake -c scanview busybox
```

(currently broken?!)

alternative:

```
cd tmp/static-scan/busybox/*/ ; python3 -m http.server
```



# CodeScanner



# CodeChecker

<https://github.com/Ericsson/codechecker>

Collection of tools to

- intercept and log the build calls
- analyse the gathered data using (clang-tidy and clangSA)
- report (static or webui)

Extension and successor of the original clang static analyser  
/ scan-build.

&lt;&gt; Code

! Issues 183

🔗 Pull requests 35

▶ Actions

📁 Projects 3

🛡 Security

📈 Insights

🔗 Branch: master ▾

Go to file

Add file ▾

↓ Code ▾



gyorb committed dbf5618 7 days ago ... ✖

🕒 3,818 commits 🔗 5 branches 🏷 36 tags

📁 .github/ISSUE_TEMPLATE	[GitHub] Fix minor grammatical things in the issue templates	7 months ago
📁 analyzer	[analyzer] Fix analyzer --file option	20 days ago
📁 bin	[license] Change license (#2729)	last month
📁 codechecker_common	Add a missing space in a debug warning	last month
📁 config	Adding new checkers to the profiles, setting severities	2 months ago
📁 docker	new dockerfiles for test environments	2 years ago
📁 docs	[tools] tu_collector get dependent source files for headers	21 days ago
📁 requirements_py/docs	Merge pull request #1935 from gyorb/readthedocs	15 months ago
📁 scripts	[license] Change license (#2729)	last month

## About

CodeChecker is an analyzer tooling, defect database and viewer extension for the Clang Static Analyzer and Clang Tidy

🔗 [codechecker.readthedocs.io](https://codechecker.readthedocs.io)

clang cpp c clang-tidy  
static-analysis linux results-viewer  
macosx codechecker llvm analysis  
database objective-c defects  
docker static-analyzer static-analyzers

📖 Readme

📄 Apache-2.0 License



Runs 5 | [Checker statistics](#) | [All reports](#) | [New features](#) x | [agl-service-gps@oneshot](#) x

[Diff](#)[Delete](#)

Diff	Name	Number of unresolved reports	Detection status	Analyzer statistics	Storage date	Analysis duration	Check command	Version tag	Description	CodeCheck er version	Delete
	agl-service-gps@oneshot	1	(1)	clangsa:  (1) clang-tidy:  (1)	2020-07-02 08:41:01	00:00:01	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
	cynagora@oneshot	17	(17)	clang-tidy:  (30) clangsa:  (30)	2020-07-02 08:00:16	00:00:35	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
	app-framework-binder@oneshot	79	(79)	clangsa:  (92)  (3) clang-tidy:  (92)  (3)	2020-07-02 07:50:44	00:02:04	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
	app-framework-main@oneshot	35	(36)	clangsa:  (34) clang-tidy:  (34)	2020-07-01 22:04:52	00:00:43	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
	agl-service-audiomixer	4	(4)	clang-tidy:  (2) clangsa:  (2)	2020-07-01 21:36:00	00:00:01	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>

# CodeChecker usage

- **Userspace tool CodeChecker is a set of python helpers**
  - main feature is that you wrap you build commands like so
    - `CodeChecker log -b "make" -o compilation.json`
  - This will preload a logger and store the compiler commands
  - With the exact commands logged, we can replay the compilation using clang and its tools clang-tidy and clangSA
    - `CodeChecker analyze compilation.json -o ./reports`

## CodeChecker usage #2


















- From there you can 'parse' into reports

- `CodeChecker parse ./reports`
- `CodeChecker parse ./reports -e html -o reports_html`

- or 'store' online in webui/frontend

- `CodeChecker store ./reports --name mypkg@v0.9 \`  
`--url http://localhost:8001/Default`

[Runs 5](#) [Checker statistics](#) [All reports](#) [New features](#) [×](#)[Diff](#)[Delete](#)

Diff	Name	Number of unresolved reports	Detection status	Analyzer statistics	Storage date	Analysis duration	Check command	Version tag	Description	CodeCheck er version	Delete
<a href="#">⊞</a> <a href="#">⊞</a>	agl-service-gps@oneshot	1	 (1)	<ul style="list-style-type: none"><li>clangsa:  (1)</li><li>clang-tidy:  (1)</li></ul>	2020-07-02 08:41:01	00:00:01	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
<a href="#">⊞</a> <a href="#">⊞</a>	cynagora@oneshot	17	 (17)	<ul style="list-style-type: none"><li>clang-tidy:  (30)</li><li>clangsa:  (30)</li></ul>	2020-07-02 08:00:16	00:00:35	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
<a href="#">⊞</a> <a href="#">⊞</a>	app-framework-binder@oneshot	79	 (79)	<ul style="list-style-type: none"><li>clangsa:  (92)  (3)</li><li>clang-tidy:  (92)  (3)</li></ul>	2020-07-02 07:50:44	00:02:04	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
<a href="#">⊞</a> <a href="#">⊞</a>	app-framework-main@oneshot	35	 (36)	<ul style="list-style-type: none"><li>clangsa:  (34)</li><li>clang-tidy:  (34)</li></ul>	2020-07-01 22:04:52	00:00:43	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>
<a href="#">⊞</a> <a href="#">⊞</a>	agl-service-audiomixer	4	 (4)	<ul style="list-style-type: none"><li>clang-tidy:  (2)</li><li>clangsa:  (2)</li></ul>	2020-07-01 21:36:00	00:00:01	<a href="#">Show</a>			6.13 (dbf5618c00 b26f41197d8 fa2f1599a37 58909924)	<input type="checkbox"/>



Runs 5 | [Checker statistics](#) | [All reports](#) | [New features](#) x | [agl-service-gps@oneshot](#) x | [cynagora@oneshot](#) x

[Bug Overview](#) | [Run history](#) | [main-cynagora-agent.c](#) x

High

L475 - core.CallAndMessage [32]

**2nd function call argument is an uninitia**

- 1 L637 - Entering loop body
- 2 L639 - Assuming the condition is tr
- 3 L677 - Assuming 'optind' is not equ
- 4 L682 - Assuming the condition is fa
- 5 L685 - Assuming 'optind' is >= 'ac'
- 6 L687 - Assuming 'piped' is 0
- 7 L699 - Assuming 'efd' is >= 0
- 8 L707 - Assuming 'rc' is >= 0
- 9 L712 - Assuming 'rc' is >= 0
- 10 L719 - Assuming 'rc' is >= 0
- 11 L726 - Assuming 'piped' is 0
- 12 L736 - Assuming 'prog' is non-null
- 13 L747 - Entering loop body
- 14 L749 - Assuming 'rc' is equal to 1
- 15 L750 - Assuming the condition is t
- 16 L751 - Assuming the condition is t
- 17 L752 - Calling 'read\_and\_dispatch'
- 18 L211 - Entered call from 'main'
- 19 L218 - Assuming 'sz' is > 0
- 20 L221 - Calling 'buf\_parse'
- 21 L163 - Entered call from 'read
- 22 L171 - Assuming 'p' is non-nu

[Show documentation](#)

Unreviewed

☒ Show arrows

Comments (0)

/home/dl9pf/AGL/codescantest/cynagora/src/main-cynagora-agent.c

Also found in: [cynagora@oneshot:main-cynagora-agent.c:L475 \[32\]](#)

```
471 if (q < 0)
472     return;
473
474 if (ac < 1 || strcmp(av[0], "sub")) {
475     reply(q, av[0], ac > 1 ? av[1] : NULL);
476 } else {
477     subquery(q, ac > 1 ? atoi(av[1]) : 1,
478             ac > 2 ? av[2] : NULL,
479             ac > 3 ? av[3] : NULL,
480             ac > 4 ? av[4] : NULL,
481             ac > 5 ? av[5] : NULL);
482 }
483 }
484
485 void dispatch_direct(int ac, char **av)
486 {
487     int q, qid;
488
489     qid = atoi(av[0]);
490     q = qidx(qid);
491     if (q < 0)
492         return;
493     dispatch(q, ac - 1, &av[1]);
494 }
```

32 < 2nd function call argument is an uninitialized value

28 < Entered call from 'read\_and\_dispatch' >

29 < Assuming 'q' is >= 0 >

30 < Calling 'dispatch' >



meta-codechecker

Cool, I want that for my builds ...



# Ok, I want CodeChecker for my OE/YP builds ...

What does the documentation say:

- <https://codechecker.readthedocs.io/en/latest/>
- There is a section about bitbake:
  - [https://codechecker.readthedocs.io/en/latest/analyzer/user\\_guide/#bitbake](https://codechecker.readthedocs.io/en/latest/analyzer/user_guide/#bitbake)

Do the following steps to log compiler calls made by [BitBake](#) using CodeChecker.

- Add `LD_LIBRARY_PATH`, `LD_PRELOAD`, `CC_LOGGER_GCC_LIKE` and `CC_LOGGER_FILE` to `BB_ENV_EXTRAWHITE` variable in your shell environment:

```
export BB_ENV_EXTRAWHITE="LD_PRELOAD LD_LIBRARY_PATH CC_LOGGER_FILE CC_LOGGER_GCC_LIKE $BB_E
```

**Note:** `BB_ENV_EXTRAWHITE` specifies an additional set of variables to allow through (whitelist) from the external environment into BitBake's datastore.

- Add the following lines to the `conf/bitbake.conf` file:

```
export LD_PRELOAD
export LD_LIBRARY_PATH
export CC_LOGGER_FILE
export CC_LOGGER_GCC_LIKE
```

- Run `CodeChecker log`:

```
CodeChecker log -o ../compile_commands.json -b "bitbake myProject"
```



Hmmm ....

Rolling up sleeves:

Maybe a blind mouldwarp like I can do something about that!

# meta-codechecker

- **Integrates Codechecker seamlessly with bitbake**
  - can write HTML reports
  - and upload to database
  - builds all necessary tools on-the-fly
    - requires meta-clang, meta-oe, meta-python

Where?: <https://github.com/dl9pf/meta-codechecker>

# meta-codechecker - Example: step-by-step

```
git clone https://github.com/kraj/meta-clang.git  
git clone https://git.openembedded.org/meta-openembedded  
git clone https://github.com/dl9pf/meta-codechecker.git
```

```
# (check the meta-codechecker'S README.md)  
git clone https://git.yoctoproject.org/git/poky  
source poky/oe-init-build-env build-test-codechecker
```

```
bitbake-layers add-layer ../meta-openembedded/meta-oe  
bitbake-layers add-layer ../meta-openembedded/meta-python  
bitbake-layers add-layer ../meta-clang  
bitbake-layers add-layer ../meta-codechecker
```

```
Next: edit conf/local.conf
```

# meta-codechecker - Example: step-by-step

```
cat << EOF >> conf/local.conf
INHERIT += "codechecker"

# disable all for now
CODECHECKER_ENABLED = "0"
# can enable _class-target for example !

# only busybox should use codechecker
CODECHECKER_ENABLED_pn-busybox = "1"

CODECHECKER_REPORT_HTML = "1"
EOF
```

# meta-codechecker - Example: step-by-step

```
bitbake busybox
```

```
tree tmp/deploy/CodeChecker/
```



# Summary



# Summary meta-clang

+++++

- meta-clang can be used by developers
- CI use possible, but needs careful list of exemptions
- straightforward workflow
- bitbake integration

-----

- documentation is good
- advanced use-cases need digging
- scanview only per-package

# Summary CodeChecker

+++++

- CodeChecker can be used by developers and in CI
- complexity hidden by pre-loaded logger library
- straightforward workflow
- parsers into multiple formats
- Webui to store and browse/review results
- bitbake integration using meta-codechecker

-----

- documentation is good, but has a few dead links and such

## Todo for meta-codechecker:

- **add easy way to inject scanner configurations**
  - e.g. select which issues to report (limit noise)
- **deal with uploading report & password or token**
- **improve recipes using pipy currently**
- **layer vs API vs CodeChecker UI dockerhub version**

## Call to action !

- Static Analysis can help improve your projects!
- Easy to use locally for development
- Integration to OpenEmbedded / Yocto Project

# Pointers / References

- meta-clang
- meta-codechecker
- [github.com/Ericsson/CodeChecker](https://github.com/Ericsson/CodeChecker)

Not covered in this talk, but highly recommended

- meta-sca



Q/A

A decorative pattern of overlapping hexagons in various shades of gray, located in the top-left corner of the slide.

# Thank you !

#dl9pf on freenode

yocto  
PROJECT

THE  
LINUX  
FOUNDATION

# Hands-on Session



# Part 1 - meta-clang (only)

# meta-clang

Start a new ssh connection / terminal.

\$>

```
source ~/yp-summit-may-21/poky/oe-init-build-env \  
~/yp-summit-may-21/poky/build-analysis-clang
```

```
bitbake-layers add-layer ~/yp-summit-may-21/src/static/meta-clang/
```

# meta-clang

```
$> rm conf/auto.conf
$> cat << EOF >> conf/auto.conf

# static analysis using clang
TOOLCHAIN = "clang"
CLANGSDK = "1"
INHERIT += "scan-build"
SCAN_BUILD = ""
SCAN_BUILD_pn-busybox = "1"
# report external server ip so we can view results ...
CLANG_SCAN_SERVER_IP = "${lwp-request -o text checkip.dyndns.org | awk '{ print $NF }'}"
EOF

$> bitbake -C unpack busybox
```

# meta-clang

```
$> bitbake -c scanview busybox
```

Open URL printed in terminal and browse around.

Hit CTRL-C to end before we continue. (2 times if necessary)

# Part 1 - meta-codechecker

# meta-codechecker

## Start a new ssh connection / terminal.

```
$> source ~/yp-summit-may-21/poky/oe-init-build-env \  
      ~/yp-summit-may-21/poky/build-analysis-codechecker  
  
bitbake-layers add-layer ~/yp-summit-may-21/src/static/meta-clang/  
  
bitbake-layers add-layer ~/yp-summit-may-21/src/static/meta-openembedded/meta-oe/  
  
bitbake-layers add-layer ~/yp-summit-may-21/src/static/meta-openembedded/meta-python/  
  
bitbake-layers add-layer ~/yp-summit-may-21/src/static/meta-codechecker/
```

# meta-codechecker

```
$> rm conf/auto.conf
$> cat << EOF >> conf/auto.conf
# static analysis using codechecker
INHERIT += "codechecker"
#e.g. enable for all target packages:
#CODECHECKER_ENABLED_class-target = "1"
# disable all
CODECHECKER_ENABLED = "0"
# only enable busybox
CODECHECKER_ENABLED_pn-busybox = "1"
# report into HTML files
CODECHECKER_REPORT_HTML = "1"
EOF
```

bitbake busybox

# check tmp/deploy/CodeChecker/busybox/ !!

```
cd tmp/deploy/CodeChecker/busybox/report-html/
python3 -m http.server 8181
```

# meta-codechecker

Step 2 ... upload to server for later inspection ...

```
$> cat << EOF >> conf/auto.conf
CODECHECKER_REPORT_STORE = "1"
CODECHECKER_REPORT_HOST = "http://ypdd.jsmo.de:8001/"
CODECHECKER_REPORT_ENDPOINT = "${hostname -s}-\${@'\${DATE}'.replace('.', '-')}"
CODECHECKER_REPORT_ENDPOINT_CREATE = "1"
EOF
```

bitbake -C **unpack** busybox

```
# check http://ypdd.jsmo.de:8001/ and locate the 'product' matching your hostname !!
#
# e.g. devday0005-20210526
```



# Notes

**Then End. Thank you !**