



# Seamless Static Analysis with Cppcheck

From IDE to CI and Code Review

DANIEL MARJAMÄKI





# Agenda

- Motivation to integrate static analysis
  - Static analysis is not used enough
  - Use Cppcheck to avoid serious vulnerabilities
- Cppcheck Integration
  - IDE
  - CI + locally
  - Other tools, i.e. code review
- Cppcheck + Clang-tidy

---

# Motivation

Not enough people use static analysis. Why?

---

# Cppcheck Debian results

- We run open source cppcheck on debian source code
  - 1000's of warnings related to undefined behavior
  - Proof that many projects are not using cppcheck
  - It is getting better



# Why don't people use static analysis?

- Usability
  - Configuration takes a bit of effort
  - Many warnings

---

## Unchecked legacy code - lots of warnings

- If you turn on all checkers at once => lots of warnings
  - Many are not serious bugs
  - There is danger to fix them
- I suggest that you only turn on a few checkers
  - Only the most critical - definite UB

---

# Why should you use Cppcheck

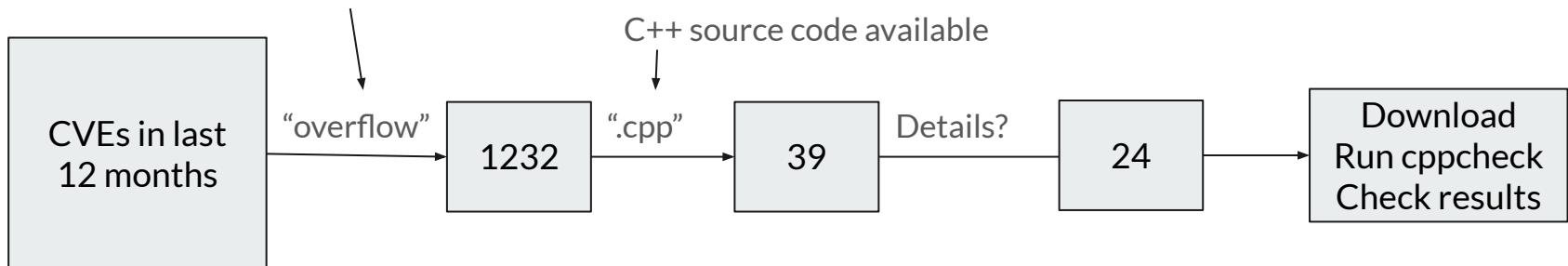
A look at security vulnerabilities

---

# Cppcheck CVE coverage

- Investigation made in march this year.
  - How many (percentage) vulnerabilities are found by Cppcheck

buffer overflow is top ranked most dangerous





# CVE coverage

CVE	Found	Kind	Info
CVE-2023-35949	Yes	format string	Found: invalidscanf
CVE-2023-35950	Yes	format string	Found: invalidscanf
CVE-2023-35951	Yes	format string	Found: invalidscanf
CVE-2023-35952	Yes	format string	Found: invalidscanf
CVE-2023-35953	Yes	format string	Found: invalidscanf
CVE-2024-0051	Premium	iterator	Found: bughuntingIteratorIncrement
CVE-2024-25580		integer overflow	integer overflow not found yet
CVE-2024-30806	Premium	pointer	Found: bughuntingBufferOverflow
CVE-2024-31002	Premium	pointer	Found: bughuntingBufferOverflow
CVE-2024-31003	Premium	pointer	Found: bughuntingBufferOverflow
CVE-2024-31580		container	Not found yet
CVE-2024-33781	Premium	pointer	Found: bughuntingBufferOverflow
CVE-2024-33782		string not zero terminated	Not found yet
CVE-2024-34408		container	Not found yet
CVE-2024-37310	Premium	integer overflow, pointer	integer overflow not found yet, dangerous pointer usage found
CVE-2024-38952	Yes	array	Found: invalidscanf
CVE-2024-39129	Yes	format string	Found: invalidscanf
CVE-2024-40658		pointer	Not found yet
CVE-2024-43091		integer overflow	integer overflow not found yet
CVE-2024-43097		integer overflow	integer overflow not found yet
CVE-2024-43767		pointer	Not found, solution: do not use raw pointer
CVE-2024-43768		integer overflow	integer overflow not found yet
CVE-2024-46426	Premium	integer overflow, pointer	integer overflow not found yet, dangerous pointer usage found
CVE-2025-25943		buffer	Not found yet

---

## Kind: format string (6, 25% of 24)

```
char header[1000];
const std::string OFF("OFF");
const std::string NOFF("NOFF");
const std::string COFF("COFF");
if(fscanf(off_file,"%s\n",header)!=1
```

(there are 6169 more such warnings, we have 1 open FP ticket)

---

# Integrating Cppcheck



# Integrating static analysis

- IDE : The earlier you get warnings the faster and easier it is to fix
  - Visual Studio Code - many
  - Clion - cppcheck plugin
  - Eclipse - cppcheckeclipse
  - Visual Studio - yes
  - Etc
- CI, locally
  - Enforcing that cppcheck is happy
- Other tools, i.e. code review



# integration: we actively work on it

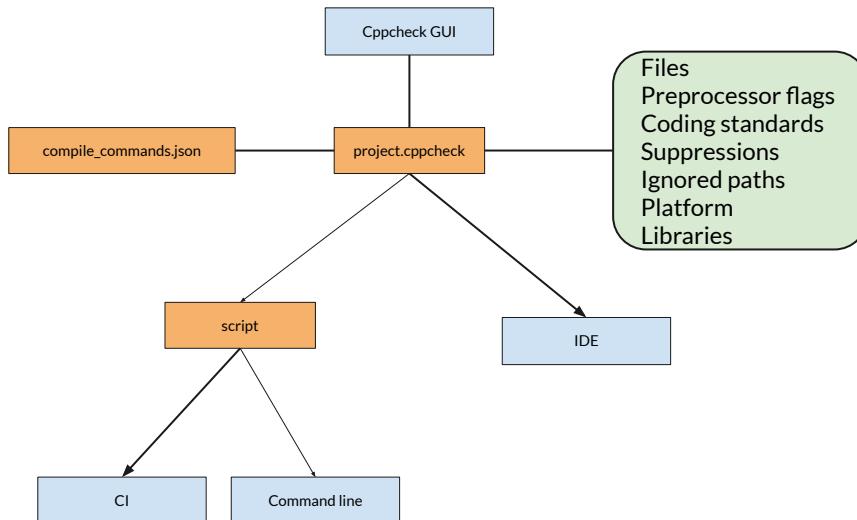
In last year:

- Added SARIF output for integration into various tools
  - Github
  - Gitlab
  - Sonarqube
  - etc
- Recently added --file-filter=+ option to simplify IDE integration
- Cppcheclipse
  - We are now maintaining this eclipse plugin
    - Thank you Konrad Windszus
  - Made it active and working again

More improvements to come..

# Using same configuration in IDE/CI/command line

---

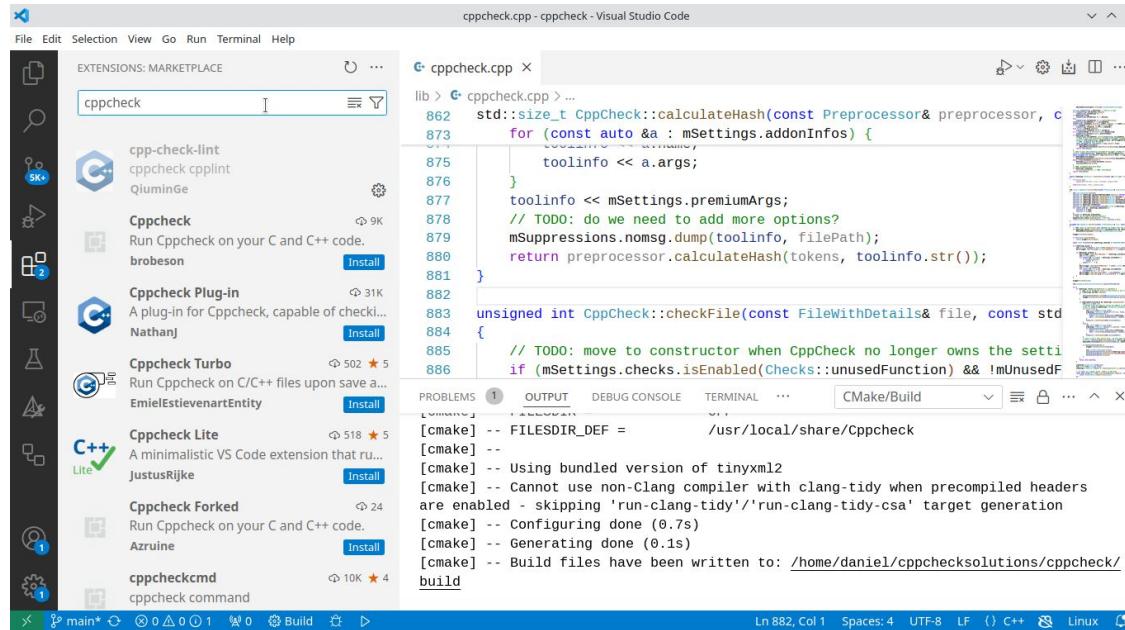


---

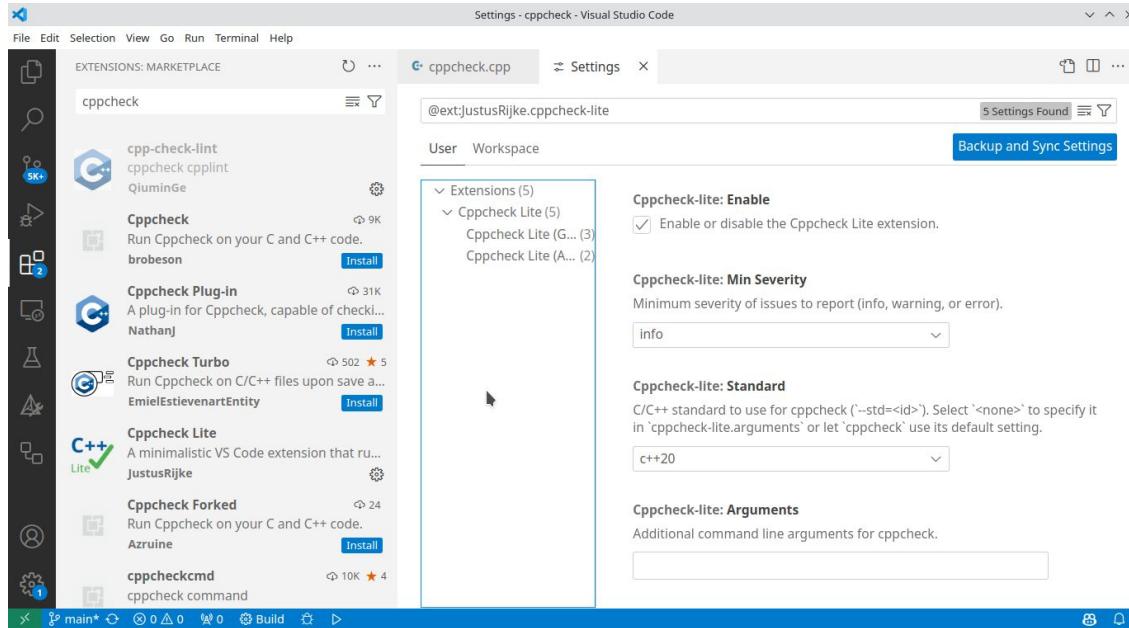
# Using a cppcheck project file

- Command line + CI:
  - Straight forward: `--project=project.cppcheck`
- IDE: Many IDE plugins do not support project files directly
  - Configure extra arguments:
    - `--project=project.cppcheck`
    - `--file-filter=+`
    - => plugin will execute such command: `cppcheck --project=project.cppcheck --file-filter=+ src/file.cpp`
      - `project.cppcheck` is loaded
      - `src/file.cpp` is checked using settings from project file, other files in the project are not checked

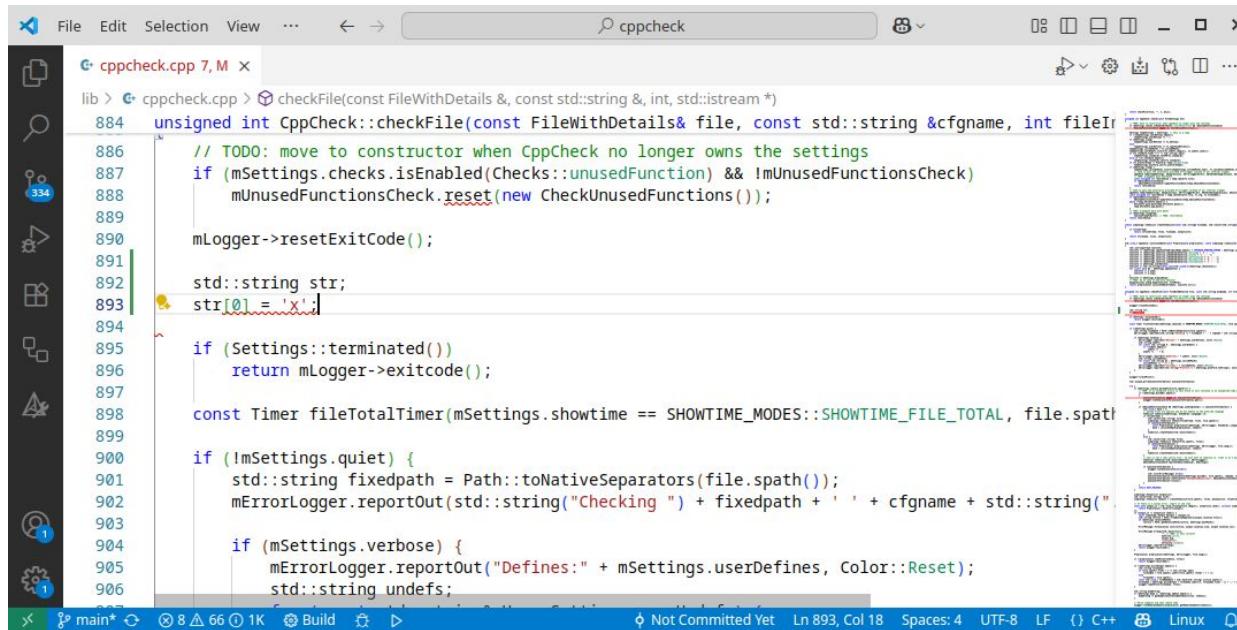
# Vscode - install cppcheck plugin



# Vscode - configuration



# Vscode - results as you type or when you save



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "cppcheck". The left sidebar contains icons for file operations like Open, Save, Find, and Settings. The main editor area displays a C++ file named "cppcheck.cpp" with line numbers 884 through 906. A red squiggly underline is under the character 'x' in the string "str[0] = 'x';", indicating a syntax error. The status bar at the bottom shows "main\* 8 66 1K Build Not Committed Yet Ln 893, Col 18 Spaces: 4 UTF-8 LF C++ Linux". On the right side, there is a large floating panel displaying a detailed list of code analysis results from the CppCheck tool, including numerous errors and warnings across multiple files.

```
lib > cppcheck.cpp 7, M x
File Edit Selection View ... ← → ⇕ cppcheck
lib > cppcheck.cpp > checkFile(const FileWithDetails &, const std::string &, int, std::istream *)
884     unsigned int CppCheck::checkFile(const FileWithDetails& file, const std::string &cfgname, int fileI
885         // TODO: move to constructor when CppCheck no longer owns the settings
886         if (mSettings.checks.isEnabled(Checks::unusedFunction) && !mUnusedFunctionsCheck)
887             mUnusedFunctionsCheck.reset(new CheckUnusedFunctions());
888
889         mLogger->resetExitCode();
890
891         std::string str;
892         str[0] = 'x';
893
894         if (Settings::terminated())
895             return mLogger->exitcode();
896
897         const Timer fileTotalTimer(mSettings.showtime == SHOWTIME_MODES::SHOWTIME_FILE_TOTAL, file.spath);
898
899         if (!mSettings.quiet) {
900             std::string fixedpath = Path::toNativeSeparators(file.spath());
901             mErrorLogger.reportOut(std::string("Checking ") + fixedpath + ' ' + cfgname + std::string(" "));
902
903             if (mSettings.verbose) {
904                 mErrorLogger.reportOut("Defines:" + mSettings.userDefines, Color::Reset);
905                 std::string undefs;
```

# Vscode - hoover to see warning

A screenshot of the Visual Studio Code (VS Code) interface. The main area shows a C++ file named "cppcheck.cpp". A tooltip is displayed over the line of code at line 884:

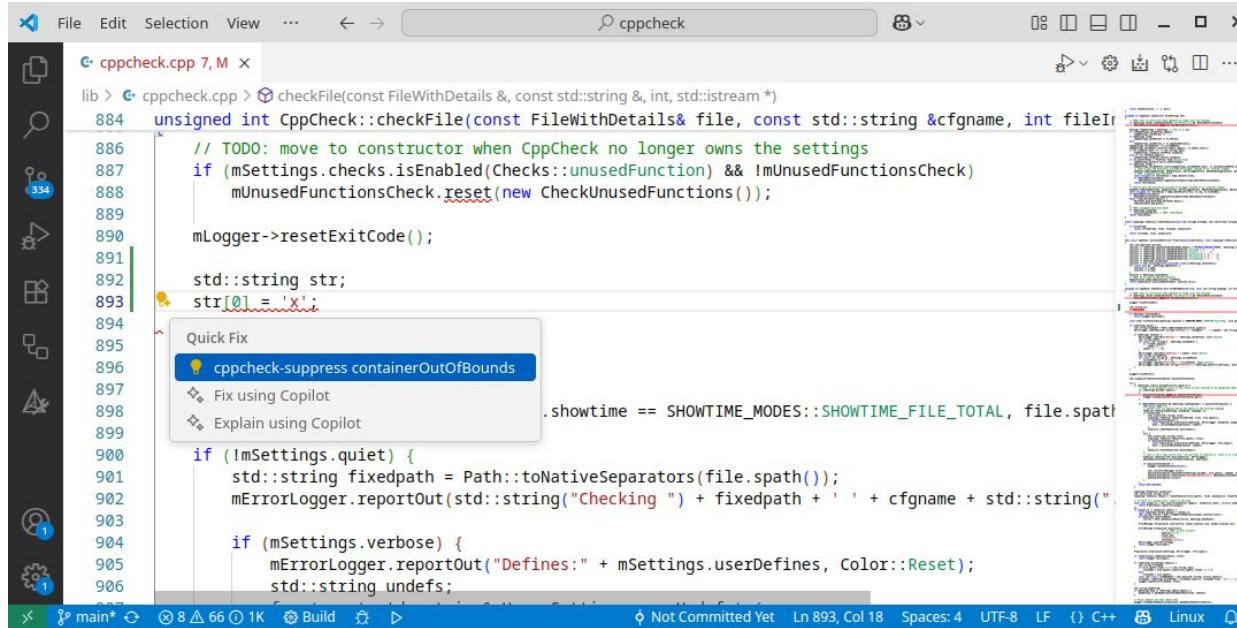
```
884     unsigned int CppCheck::checkFile(const FileWithDetails &, const std::string &, int, std::istream *)
885
886     // if (str[0] == '\0') {
887     //     mLogger->error("CWE-398 Out of bounds access in expression 'str[0]' because 'str' is empty.");
888     //     return error::containerOutOfBounds;
889     // }
890     // else {
891     //     mLogger->info("CWE-398 Out of bounds access in expression 'str[0]' because 'str' is empty.");
892     //     return error::none;
893     // }
894
895     if (Settings::terminated())
896         return mLogger->exitcode();
897
898     const Timer fileTotalTimer(mSettings.showtime == SHOWTIME_MODES::SHOWTIME_FILE_TOTAL, file.spath);
899
900     if (!mSettings.quiet) {
901         std::string fixedpath = Path::toNativeSeparators(file.spath());
902         mErrorLogger.reportOut(std::string("Checking ") + fixedpath + ' ' + cfgname + std::string(".\n"));
903
904         if (mSettings.verbose) {
905             mErrorLogger.reportOut("Defines:" + mSettings.userDefines, Color::Reset);
906             std::string undefs;
```

The tooltip contains the following information:

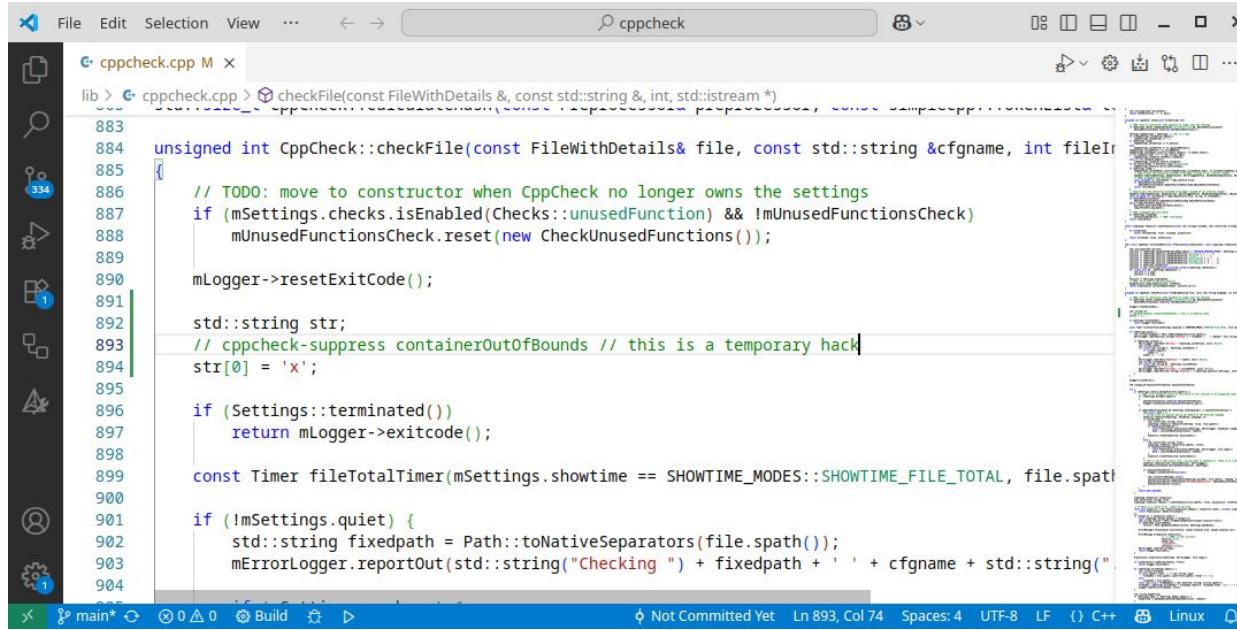
- lib > CppCheck::checkFile(const FileWithDetails &, const std::string &, int, std::istream \*)
- 884 unsigned int CppCheck::checkFile(const FileWithDetails & file, const std::string &cfgname, int fileIn)
- // if (str[0] == '\0') {
- CWE-398 Out of bounds access in expression 'str[0]' because 'str' is empty. cppcheck(error:containerOutOfBounds)
- (char)120
- mLo
- Generate Copilot summary
- View Problem (Alt+F8) Quick Fix... (Ctrl+) Fix using Copilot (Ctrl+I)

The status bar at the bottom of the screen shows the following information: main\* 8 66 1K Build Not Committed Yet Ln 893, Col 18 Spaces: 4 UTF-8 LF {} C++ Linux.

# Vscode - lightbulb options



# Vscode - suppression



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows the file name as "cppcheck.cpp". The main editor area displays the following C++ code:

```
lib > C: cppcheck.cpp > checkFile(const FileWithDetails &, const std::string &, int, std::istream *)
883
884     unsigned int CppCheck::checkFile(const FileWithDetails& file, const std::string &cfgname, int fileIndex)
885     {
886         // TODO: move to constructor when CppCheck no longer owns the settings
887         if (!mSettings.checks.isEnabled(Checks::unusedFunction) && !mUnusedFunctionsCheck)
888             mUnusedFunctionsCheck.reset(new CheckUnusedFunctions());
889
890         mLogger->resetExitCode();
891
892         std::string str;
893         // cppcheck-suppress containerOutOfBounds // this is a temporary hack
894         str[0] = 'x';
895
896         if (Settings::terminated())
897             return mLogger->exitcode();
898
899         const Timer fileTotalTimer(mSettings.showtime == SHOWTIME_MODES::SHOWTIME_FILE_TOTAL, file.spath);
900
901         if (!mSettings.quiet) {
902             std::string fixedpath = Path::toNativeSeparators(file.spath());
903             mErrorLogger.reportOut(std::string("Checking ") + fixedpath + ' ' + cfgname + std::string(" ")));
904         }
905     }
906 }
```

The code editor has several features visible: a left sidebar with icons for file operations, a search bar at the top, and a status bar at the bottom indicating "Not Committed Yet" and "Ln 893, Col 74". A large sidebar on the right displays a detailed list of code analysis results, including numerous error and warning messages from the CppCheck tool.

# Vscode - sometimes copilot can explain/fix

The screenshot shows a VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, ...
- Title Bar:** cppcheck
- Code Editor:** The file is named `cppcheck.cpp`. The code contains several TODO comments and some logic related to file processing and logging.
- Floating Panel (Copilot):**
  - Workspace:** Used 1 reference.
  - You're correct:**
  - Fix Suggestion:** A snippet of code is shown:

```
std::string str;
str[0] = 'x';
```
  - Description:** This causes undefined behavior because `str` is empty, so `str[0]` is out of bounds.
  - How to fix:** Initialize `str` with at least one character before accessing `str[0]`.
- Bottom Status Bar:** Not Committed Yet, Ln 893, Col 18, Spaces: 4, UTF-8, LF, {} C++, Linux

---

## **Integrate cppcheck results in various tools**

- SARIF ; I think this is the “future” way to export static analysis results into various tools
  - Unified interface for all your static analyzers.

# SARIF - github

---

Simple github action snippet to run cppcheck on project and upload results:

```
- name: Run Cppcheck
  run: |
    cppcheck --project=test.cppcheck --output-format=sarif 2> results.sarif

- name: Upload report
  uses: github/codeql-action/upload-sarif@v3
  with:
    sarif_file: results.sarif
    category: cppcheck
```

# SARIF - github

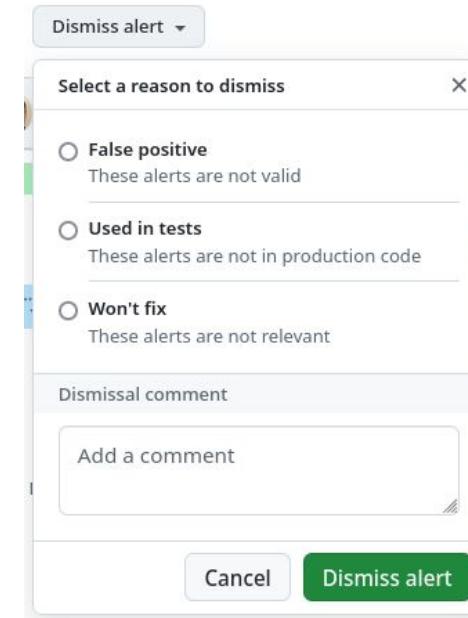
A screenshot of a GitHub pull request page. The URL is <https://github.com/danmar/cppcheck/pull/7764/files>. The pull request is titled "test #7764" and is marked as a draft. It shows 2 files changed, specifically "lib/color.cpp". The code editor displays the following snippet:

```
    72 72
    73 73 std::string toString(color c)
    74 74 {
    75 +     std::string st;
    76 +     st[0] = 'x';
    77 + }
```

The first line, "75 +", is highlighted in green, indicating it's a new line added by the check. Below the code, there are two alert sections:

**Check failure**  
Code scanning / CppCheck  
Out of bounds access in expression 'st[0]' because 'st' is empty. (Critical)  
Out of bounds access in expression 'st[0]' because 'st' is empty.  
Show more details  
Dismiss alert

**Check warning**  
Code scanning / CppCheck  
Variable 'st[0]' is assigned a value that is never used. (Warning)  
Variable 'st[0]' is assigned a value that is never used.  
Show more details  
Dismiss alert



---

# Cppcheck + clang-tidy

---

## Use other tools also

- It is a good idea to use several tools
- There is a common complaint to using several static analysis tools:
  - It means we must have several reports/plugins/scripts

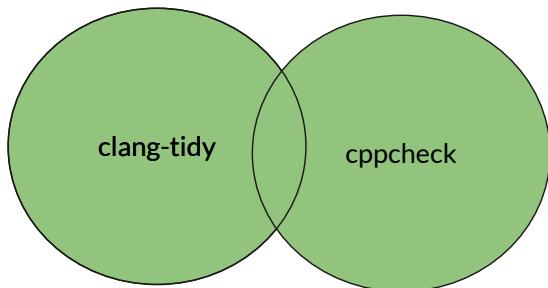
---

# Cppcheck and clang-tidy

I am a Cppcheck guy and I recommend that you use clang-tidy also

cppcheck and clang-tidy complement each other:

- Catch different bugs
- Different heuristics to catch a bug





# Configuration

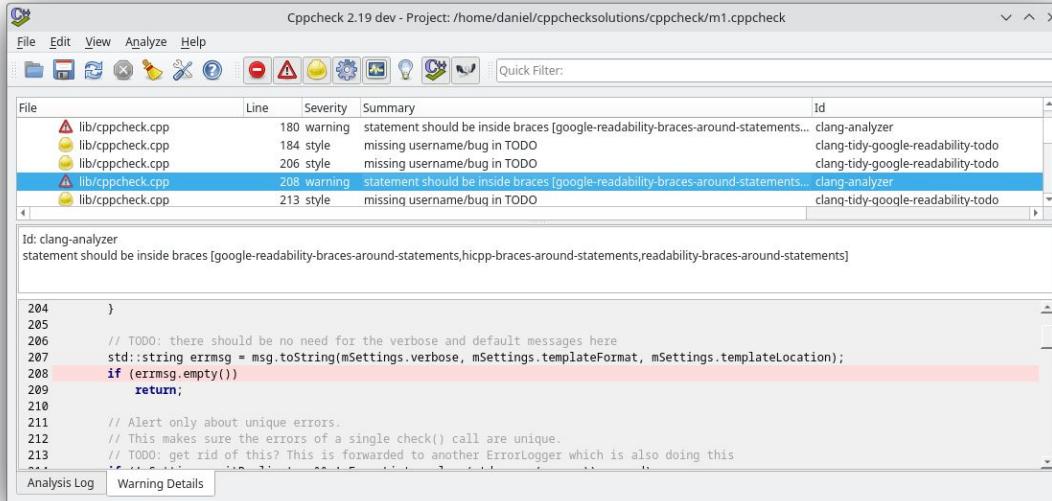
- Provide a `compile_commands.json`
- Activate clang-tidy:
  - GUI: click on option “clang-tidy” in project settings
  - Command line: `--clang-tidy`

---

## Cppcheck features for clang-tidy

- Output templates; reformatting warnings into for instance csv
- GUI
- SARIF
- Reuse same plugin/scripts
  - 1 IDE plugin
  - 1 CI plugin
  - 1 script to generate report
- Suppressions
- Multithreaded analysis
- Incremental analysis

# Clang-tidy – GUI



The screenshot shows the Cppcheck 2.19 dev interface. The menu bar includes File, Edit, View, Analyze, and Help. The toolbar contains icons for file operations like Open, Save, and Find, along with icons for code analysis and reporting. A "Quick Filter:" input field is present. The main window displays a table of analysis results:

File	Line	Severity	Summary	Id
lib/cppcheck.cpp	180	warning	statement should be inside braces [google-readability-braces-around-statements...]	clang-analyzer
lib/cppcheck.cpp	184	style	missing username/bug in TODO	clang-tidy-google-readability-todo
lib/cppcheck.cpp	206	style	missing username/bug in TODO	clang-tidy-google-readability-todo
lib/cppcheck.cpp	208	warning	statement should be inside braces [google-readability-braces-around-statements...]	clang-analyzer
lib/cppcheck.cpp	213	style	missing username/bug in TODO	clang-tidy-google-readability-todo

Below the table, a detailed view of the warning at line 208 is shown:

**Id:** clang-analyzer  
**statement should be inside braces [google-readability-braces-around-statements,hicpp-braces-around-statements,readability-braces-around-statements]**

```
204     }
205
206     // TODO: there should be no need for the verbose and default messages here
207     std::string exmsg = msg.toString(mSettings.verbose, mSettings.templateFormat, mSettings.templateLocation);
208     if (exmsg.empty())
209         return;
210
211     // Alert only about unique errors.
212     // This makes sure the errors of a single check() call are unique.
213     // TODO: get rid of this? This is forwarded to another ErrorLogger which is also doing this
```

At the bottom, tabs for "Analysis Log" and "Warning Details" are visible.



# Summary

We have talked about:

- Everybody should use static analysis
- How to integrate Cppcheck
- Cppcheck + clang-tidy