

+ 25

# Seamless Static Analysis with Cppcheck

From IDE to CI and Code Review

DANIEL MARJAMÄKI



20  
25





# 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

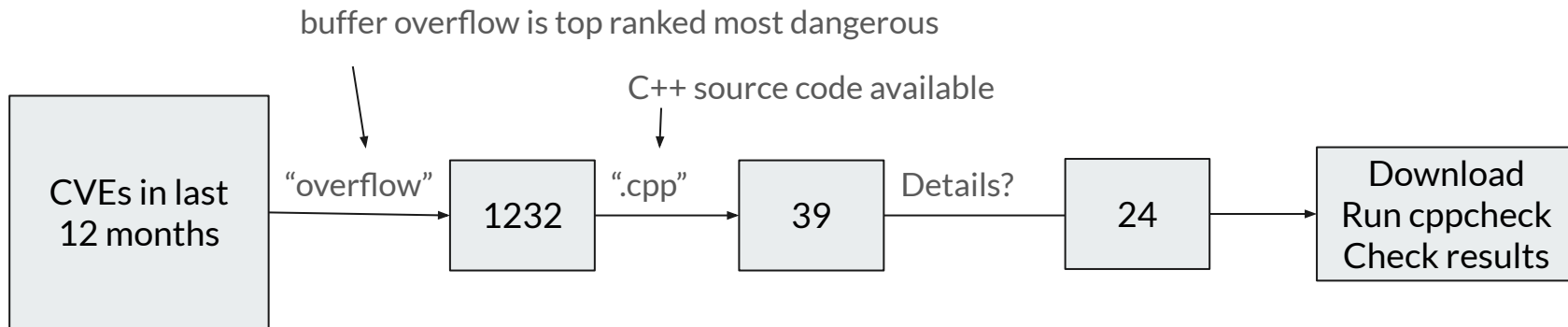


# Why should you use Cppcheck

A look at security vulnerabilities

# Cppcheck CVE coverage

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







# 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 - cppcheclipse
  - Visual Studio - yes
  - Etc
- CI, locally
  - Enforcing that cppcheck is happy
- Other tools, i.e. code review



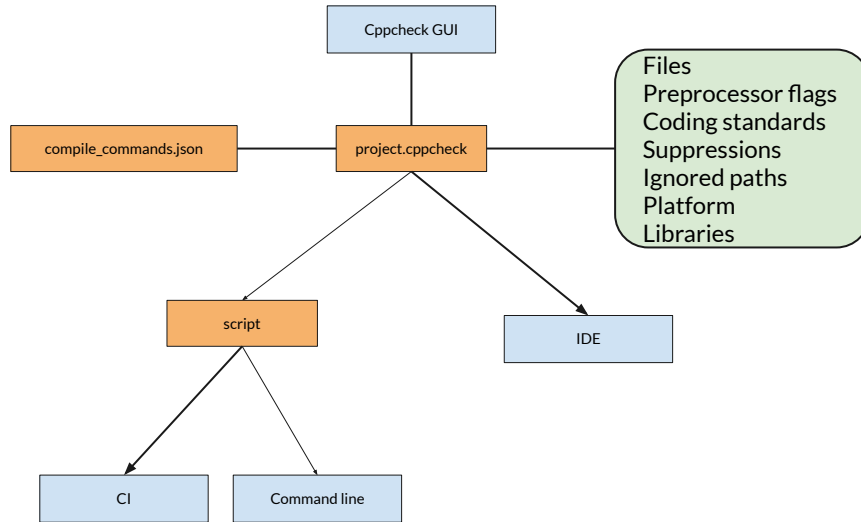
# integration: we actively work on it

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

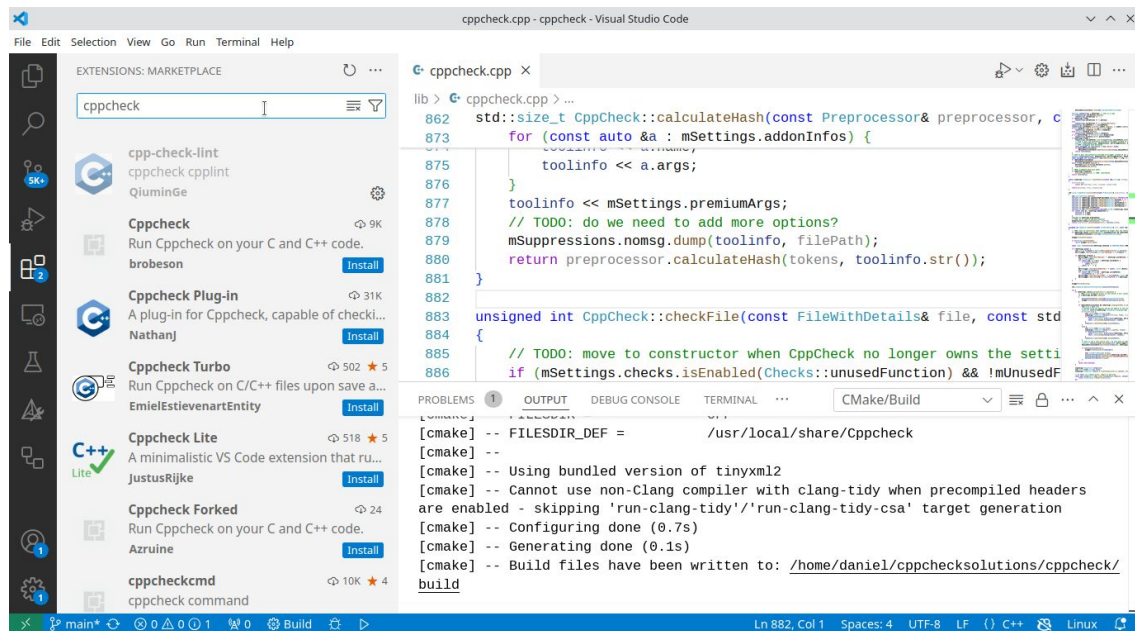




# Using a cppcheck project file

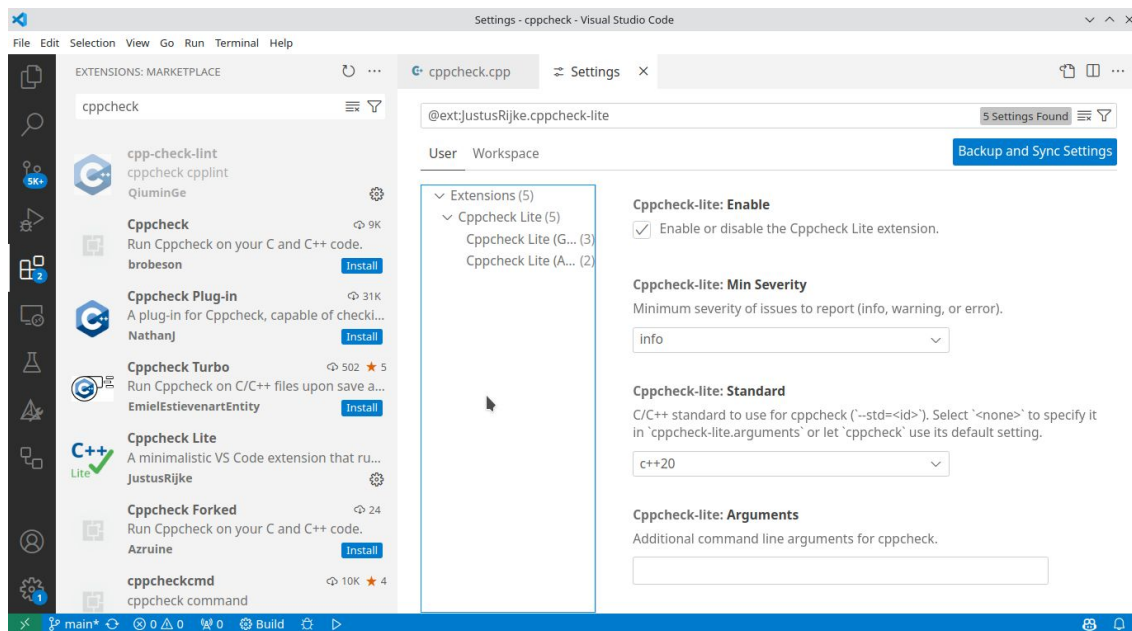
- Command line + CI:
  - Straight forward: `--project=project.cppcheck`
- IDE: Many IDE plugins do not support project files directly
  - Configure extra arguments:
    - `--project=project.cppcheck`
    - `--file-filter=+`
    - =>plugin will execute such command: `cppcheck --project=project.cppcheck --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

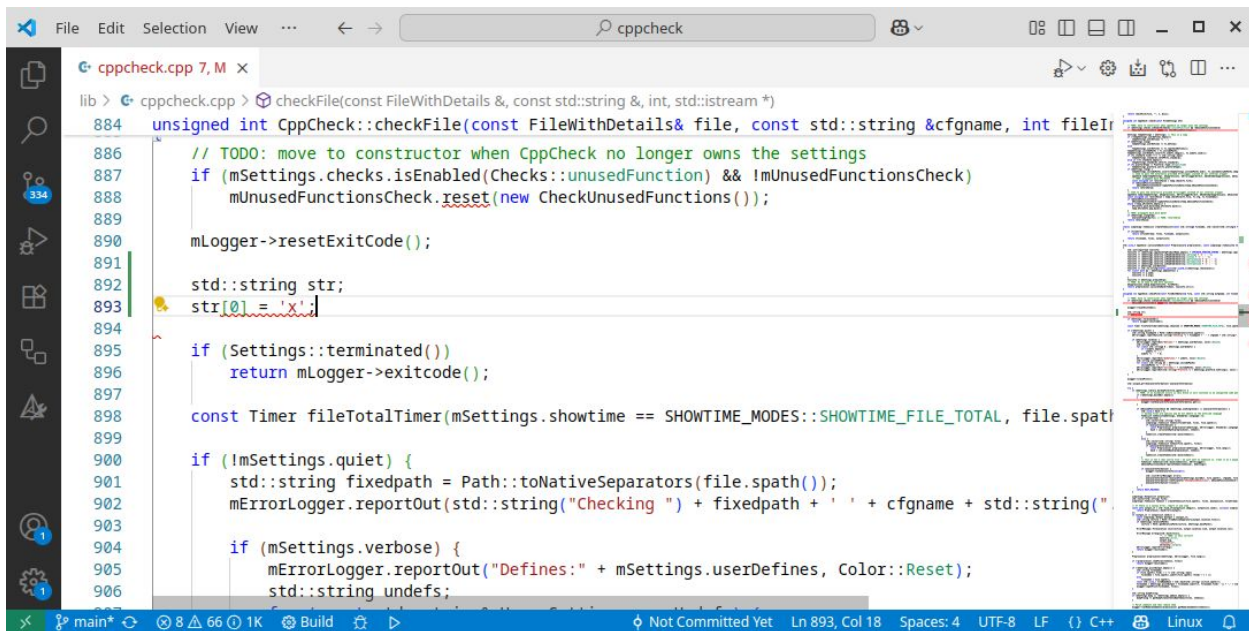




# Vscode - configuration



# Vscode - results as you type or when you save

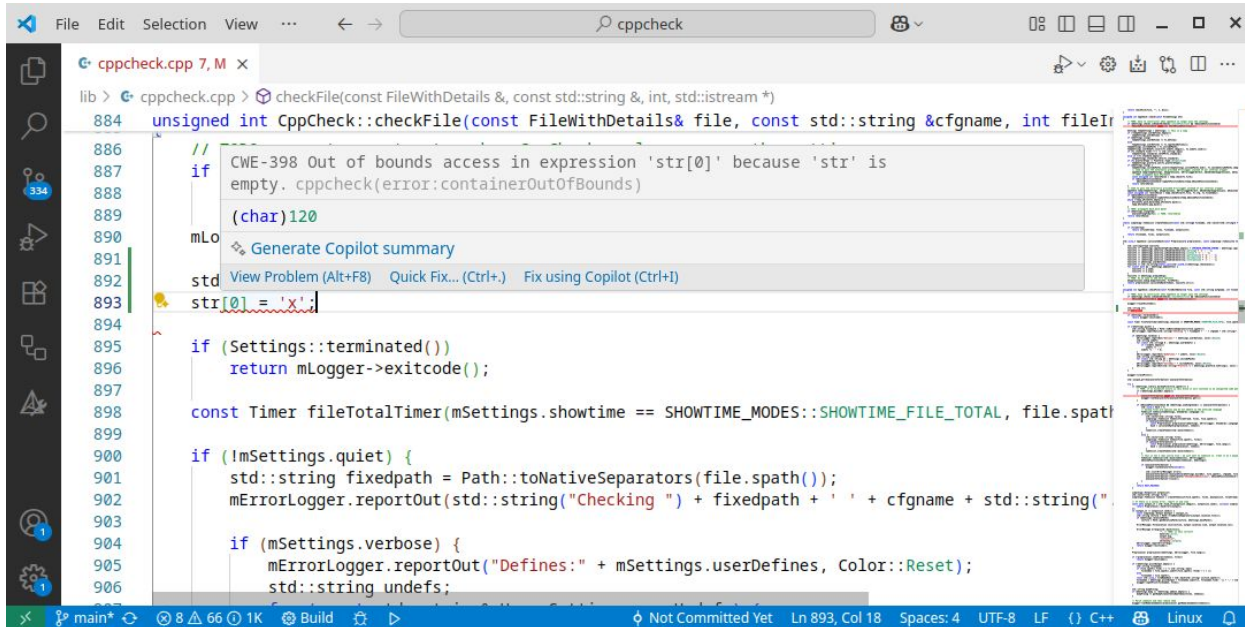


```
lib > cppcheck.cpp > checkFile(const FileWithDetails& file, const std::string&, int, std::istream*)
884 unsigned int CppCheck::checkFile(const FileWithDetails& file, const std::string& cfname, int fileIn
886 // TODO: move to constructor when CppCheck no longer owns the settings
887 if (mSettings.checks.isEnabled(Checks::unusedFunction) && !mUnusedFunctionsCheck)
888     mUnusedFunctionsCheck.reset(new CheckUnusedFunctions());
889
890 mLogger->resetExitCode();
891
892 std::string str;
893 str[0] = 'X';
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 + ' ' + cfname + std::string("
903
904     if (mSettings.verbose) {
905         mErrorLogger.reportOut("Defines:" + mSettings.userDefines, Color::Reset);
906         std::string undefs;
```

Output: cppcheck.cpp:893:1: error: stray 'X' in program

main\* 8 66 1K Build Not Committed Yet Ln 893, Col 18 Spaces: 4 UTF-8 LF C++ Linux

# Vscode - hover to see warning



The screenshot shows the Visual Studio Code editor interface. The main editor window displays a C++ file named `cppcheck.cpp`. The code is as follows:

```
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 fileIn
886 //
887 if
888     (char)120
889 mLo
890     Generate Copilot summary
891 View Problem (Alt+F8) Quick Fix... (Ctrl+.) Fix using Copilot (Ctrl+I)
892 std
893 str[0] = 'x'
894
895 if (Settings::terminated())
896     return mLogger->exitcode();
897
898 const Timer fileTotalTimer(mSettings.showtime == SHOWTIME_MODES::SHOWTIME_FILE_TOTAL, file.spat
899
900 if (!mSettings.quiet) {
901     std::string fixedpath = Path::toNativeSeparators(file.spath());
902     mErrorLogger.reportOut(std::string("Checking ") + fixedpath + ' ' + cfgname + std::string("
903
904     if (mSettings.verbose) {
905         mErrorLogger.reportOut("Defines:" + mSettings.userDefines, Color::Reset);
906         std::string undefs;
```

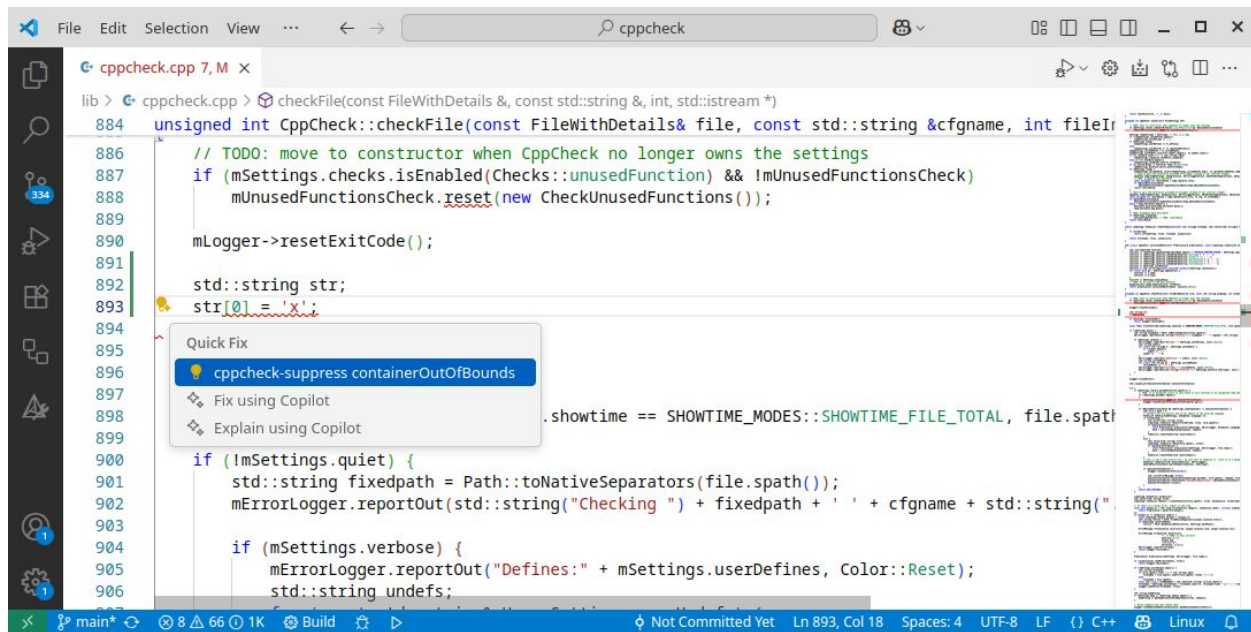
A warning tooltip is displayed over the line `str[0] = 'x';`. The tooltip contains the following text:

CWE-398 Out of bounds access in expression 'str[0]' because 'str' is empty. cppcheck(error:containerOutOfBounds)

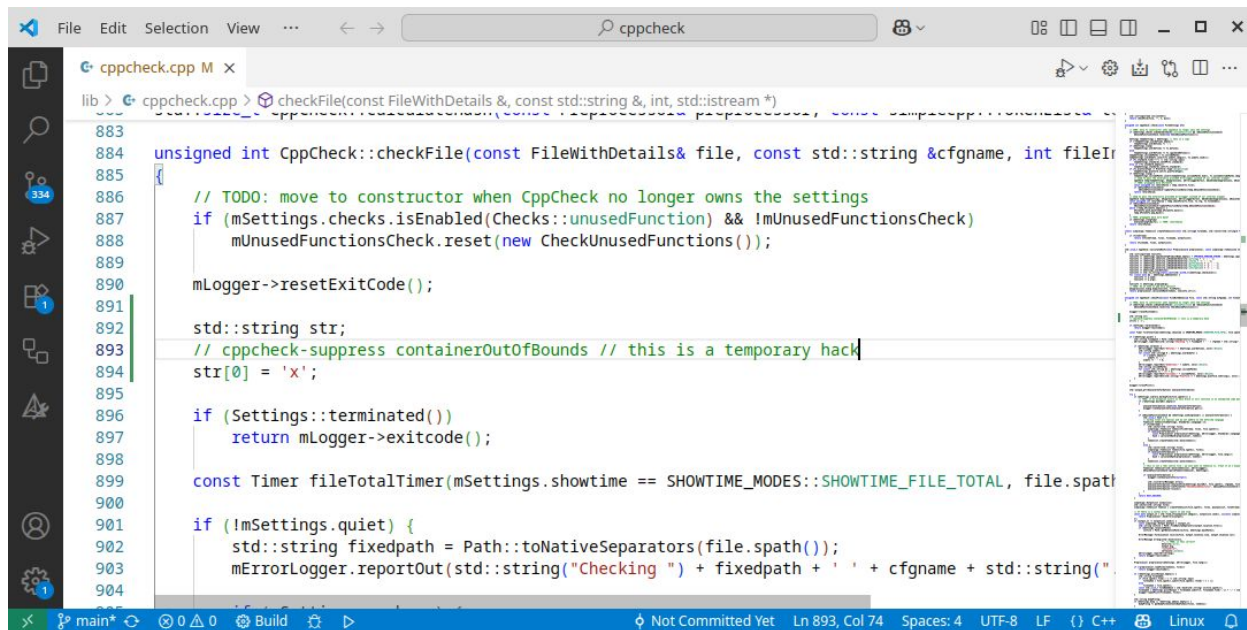
The status bar at the bottom of the editor shows the following information:

- main\*
- 8 66 1K
- Build
- Not Committed Yet
- Ln 893, Col 18
- Spaces: 4
- UTF-8
- LF
- {}
- C++
- Linux

# Vscode - lightbulb options



# Vscode - suppression

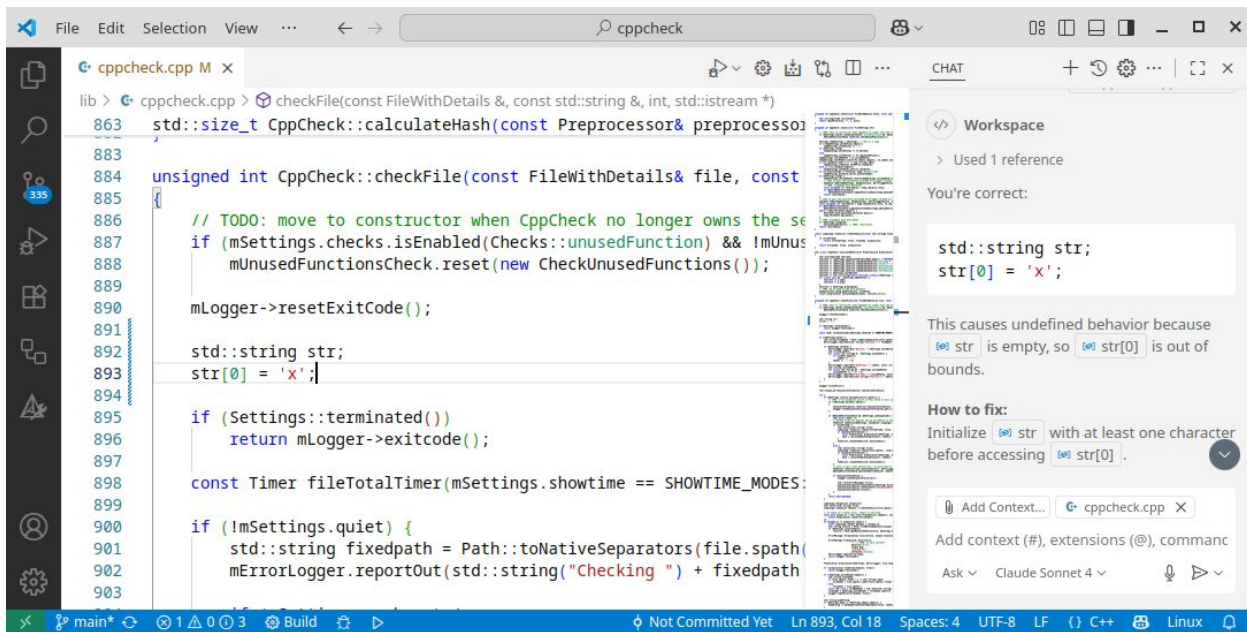


The screenshot shows the Visual Studio Code editor interface. The main editor window displays a C++ file named `cppcheck.cpp`. The code is as follows:

```
lib > cppcheck.cpp > checkFile(const FileWithDetails& file, const std::string &cfgname, int fileIn
883
884 unsigned int CppCheck::checkFile(const FileWithDetails& file, const std::string &cfgname, int fileIn
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
```

The comment on line 893, `// cppcheck-suppress containerOutOfBounds // this is a temporary hack`, is highlighted in green. The status bar at the bottom indicates the file is not committed, the cursor is at line 893, column 74, and the file is encoded in UTF-8 with LF line endings.

# Vscode - sometimes copilot can explain/fix





# 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

The screenshot shows a GitHub pull request interface for a repository named 'danmar / cppcheck'. The pull request is titled 'test #7764' and is in a 'Draft' state. It shows a merge from 'danmar:main' to 'cppcheck:solutions:test-github-sarif'. The interface includes tabs for 'Code', 'Pull requests', 'Actions', 'Security', 'Insights', and 'Settings'. The 'Pull requests' tab is active, showing the pull request details. Below the pull request title, there is a section for 'Files changed' showing a diff of the file 'lib/color.cpp'. The diff shows changes to the 'toString' function. Below the diff, there are two SARIF alerts from CppCheck. The first alert is a 'Check failure' with a 'Critical' severity, stating 'Out of bounds access in expression 'st[0]' because 'st' is empty.' The second alert is a 'Check warning' with a 'Warning' severity, stating 'Variable 'st[0]' is assigned a value that is never used.'

test #7764

danmar wants to merge 1 commit into danmar:main from cppcheck:solutions:test-github-sarif

Conversation 4 Commits 1 Checks 63 Files changed 2

Filter changed files

lib

- color.cpp
- cppcheck.cpp

lib/color.cpp

```
72 72 @@ -72,6 +72,10 @@
73 73 std::string toString(Color c)
74 74 {
75 75 +
76 76 + std::string st;
77 77 + st[0] = 'x';
```

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

Reply...

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

The dialog box is titled 'Dismiss alert' and contains a 'Select a reason to dismiss' section. There are three radio button options: 'False positive' (These alerts are not valid), 'Used in tests' (These alerts are not in production code), and 'Won't fix' (These alerts are not relevant). Below these options is a 'Dismissal comment' section with a text input field labeled 'Add a comment'. At the bottom of the dialog are two buttons: 'Cancel' and 'Dismiss alert'.

Dismiss alert

Select a reason to dismiss

☐ False positive  
These alerts are not valid

☐ Used in tests  
These alerts are not in production code

☐ Won't fix  
These alerts are not relevant

Dismissal comment

Add a comment

Cancel 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

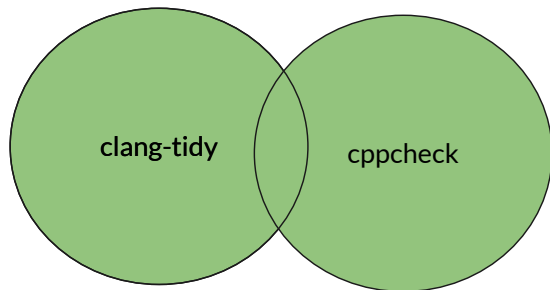


# Cppcheck and clang-tidy

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

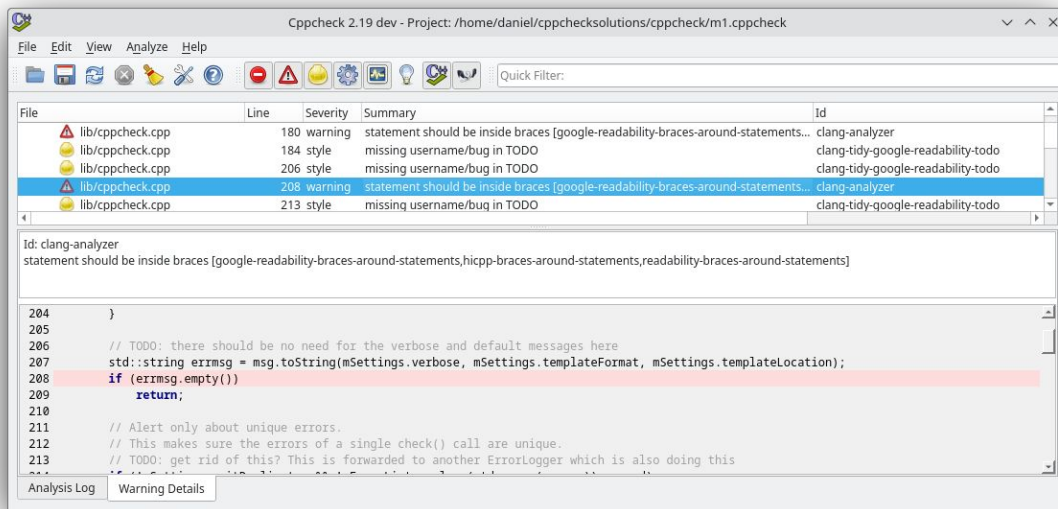
- Provide a compile\_commands.json
- Activate clang-tidy:
  - GUI: click on option “clang-tidy” in project settings
  - Command line: --clang-tidy



## 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





# Summary

We have talked about:

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