

P1706R3: Programming Language Vulnerabilities for Safety Critical C++

Date: 2021-04-15 (April 2021 mailing): 10 AM ET

Project: ISO JTC1/SC22/WG21: Programming Language C++

Audience: SG12, WG21, WG23, MISRA

Authors: Michael Wong (Codeplay), Stephen Michel(WG23), Peter Sommerlad,

Contributors: SG12: WG23, Lisa Lippincott, Aaron Ballman, Richard Corden, Clive Pygott, Erhard Ploedereder, John McFarlane, Paul Preney, Andreas Weis, Federico Kircheis, Tullio Vardanega, Jorg Brown, Chris Tapp

Emails: michael@codeplay.com

Reply to: michael@codeplay.com

Introduction

This document describes the continuing work of reviewing MISRA and WG23 Programming Vulnerabilities in C++ document in WG21, in SG12.

Revision History

R3: 2021-04-15 this revision: covers work since R2 including move Github, and more clause updates

R2: 2020-01-13 (pre-Prague) revision, covers work from Belfast where we considered MISRA rules as well as several additional WG23 rules

R1: 2019-10-07 (Belfast mailing) detailing work done so far in collaboration with WG23 and intention to work with MISRA

R0: initial version describing charter with WG23 and work status links

Motivation and Background

WG23 is an ISO WG under SC22 that looks at programming vulnerabilities of various languages. Since 2017, WG23 has requested a liaison with WG21 to closely collaborate on their work of documenting programming vulnerabilities for C++. This was shown in a presentation [N0729] in the July 2017 WG23 meeting where the background and history of WG23 was presented to SG12. A request was made to WG21 to lead with the documentation of

vulnerabilities in C++, so as to ensure their accurate representation using WG21's technical expertise.

This was accepted using a co-located meeting format where WG23 members was to attend WG21 meetings. SG12's charter was expanded from Undefined Behavior and Unspecified Behavior to add Vulnerabilities to the title

SG 12 agreed that WG 21 needs to do something about vulnerabilities. It was pointed out that even Ada, widely acknowledged to be a "safe" language, acknowledges 50 of the 63 language-defined vulnerabilities. The goals of this is to work together and not work to make C++ look bad. We are not specifying subsets or what language features to avoid, simply pointing out how vulnerabilities occur and giving guidance. [N0730]

Since then, we have added MISRA C++ as participants in the review process.

.
As a result these co-located meetings have been occurring in SG12 in every WG21 meeting. This is a report of its progress.

At the Belfast meeting, we proceeded with examining the MISRA C++ draft in a co-located manne and will continue in Prague.

At the Prague meeting, we continued with processing additional clauses.

In November 2020, Peter Sommerlad and Paul Preney moved the document from Word to Github.

Impact on the Standard

The result of this is a WG23 document that specifically updates on C++ Programming Vulnerabilities guidelines, along with those from Ada, C, Fortran, etc. These are most of the languages under SC22. We have also been feeding back new findings from this joint WG23/SG12 meeting back to update Core Guidelines.

It will also result in a MISRA C++ updated document that tracks more closely to C++14/17/20 and future revisions.

The goal is to have an active group reviewing various Safety and Vulnerabilities documents and improve them for their accurate portrayals. In future, this will include MISRA C++/AUTOSAR C++ guidelines which is also actively revising their documents. As the various Standards change, we will also need to continue updates. This is still a WIP.

Github

In November 2020, Peter Sommerlad and Paul Preney moved the document from Word to Github and Markdown format. HTML and Word format is generated with pandoc. There are two github repositories with the content of the original word document and further progression.

You can view the full document as well as individual pages at

<https://iso-iec-jtc1-sc22-wg23-cpp.github.io/wg23-tr24772-10-public/>

The editing takes place in markdown on the closed-user group repository

<https://github.com/ISO-IEC-JTC1-SC22-WG23-CPP/wg23-tr24772-10>

WG23 member and WG21/SG12 members can contact Paul or Peter to be added to the editors of this repository. A github user account is necessary.

The content of <https://github.com/ISO-IEC-JTC1-SC22-WG23-CPP/wg23-tr24772-10-public> is generated through a github action on the private repository that pushes it to the public version. It also contains a generated word document.

Status

MISRA

At Belfast meeting, we had a full meeting with MISRA which we plan to continue in Prague. The minutes are here:

N0904: [Meeting notes from WG 21/SG 12 meeting with MISRA C++](#)

At the Prague meeting, we discussed:

- Single Exit
- Banning Raw pointers
- Brace Initialization
- Exceptions

The notes are:

<https://wiki.edg.com/bin/view/Wg21prague/SG12>

At this point, MISRA is proceeding with a proposed release and update of the MISRA 2008 Standard which was based on C++98, with the intention to update to C++14/17. We discussed issues of:

- Example of single exit
- Example of dead code vs unused code
- Braced-initialization {}
- two rules that conflict:
 - Every switch will have a default
 - No unreachable code
- We propose a single rule that says that every value of an enum must be covered by a switch alternative (no default)

WG23

In the fall of 2020, ISO/IEC 24772-10 was moved into GitHub. The group has been meeting every 4 weeks to discuss open issues, propose revisions, and to progress work

As of 12 April 2020, 53 issues are open (5 of which have outstanding pull requests to be reviewed) and 17 have been closed in the GitHub repository. The document can be accessed at the GitHub link.

Future meeting dates in 2021

(Every 3 weeks from June 7)

April 12, May 10, June 7, June 28, July 19, August 9, August 30, September 20, October 11, November 1, November 22, December 13

Since July 2017, we have held regular meetings. These are the documents that we have generated in WG23. All are located in the WG23 repository:

<http://www.open-std.org/JTC1/SC22/WG23/docs/documents>

Since Prague, WG23 has been meeting monthly through 2020 to today and plans to continue. The meetings are in the C++ calendar under SG12.

The latest WG23 progress since the Belfast meeting is here:

N0908: [TR 24772-10](#) [C++ language vulnerabilities after meeting 66 with WG 21/SG 12](#)

For N0908, the following clauses are essentially completed.

- 6.2 type system

- 6.3 Bit representation
- 6.4 Floating Point
- 6.5 Enumerator issues [CCB],
- 6.6 Conversion errors
- 6.7 String termination
- 6.8 Buffer boundary violation
- 6.9 Unchecked array indexing
- 6.10 Unchecked array copying (needs to be revisited)
- 6.11 Pointer type conversions
- 6.12 Pointer arithmetic
- 6.13 Null pointer dereference [XYH],
- 6.14 Dangling reference to heap
- 6.15 Arithmetic wrap-around error
- 6.16 Using shift operations for multiplication and division
- 6.17 Choice of clear names [NAI]
- 6.18 Dead Store
- 6.19 Unused variables
- 6.20 Identifier name reuse
- 6.21 Namespace Issues
- 6.22 Initialization of variables [LAV]
- 6.23 Operator precedence and associativity
- 6.24 Side effects and order of evaluation
- 6.25 Likely incorrect expression
- 6.26 Dead store,
- 6.27 Switch statements and static analysis
- 6.28 Demarcation of control flow
- 6.29 Loop control variables
- 6.30 Off-by-one errors
- 6.31 Structured programming

- 6.32 Passing parameters and return values
- 6.33 Dangling references to stack frames
- 6.34 Subprogram signature mismatch
- 6.35 Recursion
- 6.36 Ignored error status and unhandled exceptions
- 6.37 Type breaking reinterpretation of data
- 6.38 Deep vs shallow copying [YAN]
- 6.39 Memory leak and heap fragmentation
- 6.41 Inheritance
- 6.42 Violations of the Liskov substitution principle
- 6.43 Redispatching
- 6.44 Polymorphic variables
- 6.45 Extra intrinsics
- 6.46 Argument passing to library functions
- 6.47 Inter-language calling
- 6.48 Dynamically-linked code and self-modifying code [NYY]
- 6.49 Library Signature
- 6.50 Unanticipated exceptions from library routines
- 6.51 Pre-processor directives
- 6.52 Suppression of language-defined run-time checking
- 6.53 Provision of inherently unsafe operations
- 6.54 Obscure language features
- 6.55 Unspecified behaviour
- 6.56 Undefined behaviour
- 6.57 Implementation-defined behaviour
- 6.58 Deprecated language features
- 6.59 Concurrency -- Activation
- 6.60 Concurrency – Directed termination
- 6.64 Uncontrolled format string

Ongoing in GitHub, as well as open issues.

- 6.40 Templates and generics
- 6.61 Concurrent data access
- 6.62 Concurrency – Premature termination
- 6.63 Protocol lock errors

Acknowledgements

This work benefits from Wg23, SG12, MISRA, and Autosar members.

Michael Wong's work is made possible by Codeplay Software Ltd., ISOCPP Foundation, Khronos and the Standards Council of Canada.

References

[N0729]

http://www.open-std.org/JTC1/SC22/WG23/docs/ISO-IECJTC1-SC22-WG23_N0729-presentation-WG21-programming-language-vulnerabilities-20170713.ppt

[N0730]

http://www.open-std.org/JTC1/SC22/WG23/docs/ISO-IECJTC1-SC22-WG23_N0730-report-from-SC22-WG21-SG12-meeting-20170713.docx

[N0766]

http://www.open-std.org/JTC1/SC22/WG23/docs/ISO-IECJTC1-SC22-WG23_N0766-possible-liaison-statement-WG23-WG21-SG12.zip

[N0885] [TR 24772-10 C++ language vulnerabilities after WG 23 meeting 63 with edits by S. Mitchell](#)