

# P1706R1: Programming Language Vulnerabilities for Safety Critical C++

**Date:** 2019-10-07 (Belfast mailing): 10 AM ET

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

**Audience:** SG12, WG21, WG23

**Authors:** Michael Wong (Codeplay)

**Contributors:** SG12: Stephen Michel, Peter Sommerlad, Lisa Lippincott, Aaron Ballman, Richard Corden, Clive Pygott, Erhard Ploedereder, John McFarlane, Paul Preney, Andreas Weis, Federico Kircheis, Tullio Vardanega, Jorg Brown,

**Emails:** [michael@codeplay.com](mailto:michael@codeplay.com)

**Reply to:** [michael@codeplay.com](mailto:michael@codeplay.com)

## Introduction

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

## Revision History

R1: this update 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, 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 basically 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]

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

There is now recent work towards an exploratory group for C++ Safety Critical in CPPCON 2019[cppcon].

At the Belfast meeting, we intend to also proceed with examining MISRA C++ draft in a co-located manner.

## 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 joining WG23/SG12 meeting back to update Core Guidelines.

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++ as well as 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.

## Proposals

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>

The latest WG23 progress since the Cologne meeting is here:

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

At this point in time, the following clauses are essentially completed first pass.

- 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

- 6.25 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

TBD

- 6.2 Type System
- 6.4 Floating point
- 6.20 Identifier name reuse
- 6.24 Side effects and order of evaluation
- 6.40 Templates and generics
- 6.61 Concurrent data access
- 6.62 Concurrency – Premature termination
- 6.63 Protocol lock errors
- 6.64 Uncontrolled format string

## Acknowledgements

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](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](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](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. Michell](#)

[cppcon] Bryce Lebach cpp-safety-critical google group; Private Communication.