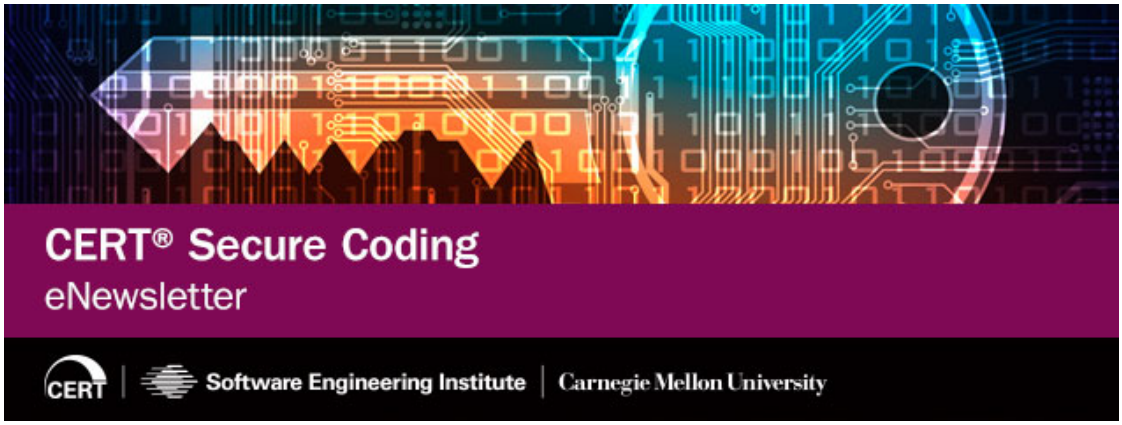


**Summer Edition 2016**

---



## Welcome to the Summer 2016 Edition of the CERT Secure Coding Standards eNewsletter!

Another season is upon us, and we have been busy with the upcoming and recent events and activities described in this newsletter. We hope you had a great summer and are getting ready for the fall (and new fiscal year for some).

## Register Now!

Our Secure Coding Symposium is scheduled for 8 September 2016 in the Arlington, VA area. We will have great speakers from government, industry, and academia who will talk about future trends affecting Secure Coding, including keynotes from Dr. Peter Fonash of DHS and Mary Ann Davidson of Oracle. Registration for the symposium is still open; there are fewer than 20 seats still available!

## C++ Standard Reviewers Needed

The SEI CERT C++ Coding Standard is entering the final technical review stage before its pending publication later this Fall. Anyone interested in the content of the standard should review topics and mention any issues or suggestions as a comment on the rules pages so that we can address all issues and respond to questions. We are focusing on the rules (not the recommendations) at this time.

We plan to publish the CERT C++ Coding Standard as a free PDF, similar to the release of the SEI CERT C Coding Standard. We appreciate all help in making sure that the standard reflects the best practices of the community. All constructive contributors will be recognized in the standard when published.

### CERT Secure Coding in Java Professional Certificate Released

A few weeks ago, we released the CERT Secure Coding in Java Professional Certificate. It joins the CERT Secure Coding in C and C++ Professional Certificate that was released earlier this year. They are both eLearning professional development offerings. Each Professional Certificate teaches secure software concepts and secure coding best practices for the respective programming language.

In these certificate programs, students take two courses of self-paced, online material and complete a required examination. The certificate programs are especially helpful for providing secure coding training for very large groups of developers, or for individuals or small teams of developers. We also offer instructor-led training at customer sites, designed for groups of 15-30 per delivery session.

## Upcoming Events

Don't forget to register for our Secure Coding Symposium on 8 September, 2016 in the Arlington VA area.

CERT, the Software Engineering Institute, and Carnegie Mellon University are hosting the upcoming ISO WG14/PL22.11 C Standard meeting in Pittsburgh on 17-21 October 2016. Several members of our team will be participating, including Dan Plakosh, Aaron Ballman, and David Svoboda.

Lori Flynn is chair of the SPLASH co-hosted workshop, Mobile! 2016, which will take place 31 October, 2016 in Amsterdam, The Netherlands. Please consider submitting a paper or just attending this workshop on mobile application development and analysis.

David Svoboda will give three presentations at JavaOne 2016 in September:

- Inside the CERT Oracle Secure Coding Standard for Java
- Exploiting Java Serialization for Fun and Profit
- The Java Security Architecture: How and Why

The following papers and tutorials have been accepted to the Security Development Conference in November:

- Static Analysis Alert Audits: Formal Lexicon & Rules by David Svoboda, Lori Flynn, and Will Snavely
- Automated Code Repair Based on Inferred Specifications by William Klieber
- *Tutorial: Beyond errno: Error Handling in C* by David Svoboda

## Recent Events

Mark Sherman presented "Construction and Implementation of CERT Secure Coding Rules Improving Automation of Secure Coding" at the Safe and Secure Systems and Software Symposium (S5) on 13 July 2016. The presentation was co-developed by Mark Sherman and Aaron Ballman.

Mark Sherman presented "Risks in the Software Supply Chain," at Abstractions on 18-20 August 2016 in Pittsburgh, PA.

Mark Sherman and Bob Schiela presented the webinar From Secure Coding to Secure Software on 17 August 2016.

## SEI CERT Secure Coding Standard Updates

### CERT C Coding Standard

Editors: Aaron Ballman, SEI/CERT

Martin Sebor, Red Hat, Inc.

Download the latest stable version.

***Changed***

- CON00-C is now a rule: CON43-C. Do not allow data races in multithreaded code.

*No C rules were added or removed.*

**New Clang Checkers**

### CERT C++ Secure Coding Standard

Editors: Aaron Ballman, SEI/CERT

Martin Sebor, Red Hat, Inc.

***Added***

- CON56-CPP. Do not speculatively lock a non-recursive mutex that is already owned by the calling thread
- EXP64-CPP. Do not call a function with a mismatched language linkage

***Changed***

- CON50-CPP. Do not destroy a mutex while it is locked

  Extended the rule to cover `std::mutex`, `std::recursive_mutex`, `std::timed_mutex`, `std::recursive_timed_mutex`, and `std::shared_timed_mutex`.
- CTR57-CPP. Provide a valid ordering predicate

  Corrected the transitivity requirement wording.
- MEM51-CPP. Properly deallocate dynamically allocated resources

Added an NCCE/CS pair for `std::shared_ptr` holding an array, which requires slightly different treatment than `std::unique_ptr` in its compliant solution.

- DCL50-CPP. Do not define a C-style variadic function

Corrected the NCCE to properly pass the argument before the ellipsis to `va_start()`.

- DCL57-CPP. Do not let exceptions escape from destructors or deallocation functions

Corrected several issues with the code examples.

- DCL59-CPP. Avoid information leakage when passing a class object across a trust boundary

Corrected the last CS to not use a variable-length array (which is not a C++ feature).

- EXP52-CPP. Do not rely on side effects in unevaluated operands
- Corrected the code example for exception #1; a line continuation character was missing.
- EXP62-CPP. Do not access the bits of an object representation that are not part of the object's value representation

Corrected a typo with the rule's exception.

- MEM50-CPP. Do not access freed memory

Corrected a typo with the `std::string::c_str()` compliant solution.

- MEM54-CPP. Provide placement new with properly aligned pointers to sufficient storage capacity

Added missing includes and a declaration for `S` to the last compliant solution.

- ERR53-CPP. Do not reference base classes or class data members in a constructor or destructor function-try-block handler

Removed a NCCE/CS pair that showed a valid NCCE but an invalid CS for which no valid CS was possible.

- ERR55-CPP. Honor exception specifications

Corrected code examples to be self-contained so that they compile.

- CON53-CPP. Avoid deadlock by locking in a predefined order

Corrected the last CS to return success or error rather than letting control reach the end of a non-`void` function.

- CON54-CPP. Wrap functions that can spuriously wake up in a loop

Renamed identifiers to not end with `_t`, which are reserved for POSIX implementations.

- DCL56-CPP. Avoid cycles during initialization of static objects

Expanded the rule to cover dynamic initialization interdependencies and renamed the rule (was previously called DCL56-CPP. Do not recursively reenter a function during the initialization of one of its static objects).

- STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator

Removed a NCCE that didn't really add a lot of value, improved the last CS so that it no longer relies on the input having exactly 32 characters.

- CTR54-CPP. Do not subtract iterators that do not refer to the same container

Corrected the first NCCE to properly handle the case where the tested pointer precedes the given range.

- CTR52-CPP. Guarantee that library functions do not overflow

Changed the title (was previously called CTR52-CPP. Guarantee that library functions do not form invalid iterators) and added another NCCE/CS pair.

***Removed***

- DCL57-CPP. Functions declared with [[noreturn]] must return void

Was demoted from a rule; is now a recommendation. Currently listed as DCL22-CPP. Functions declared with [[noreturn]] must return void.

**New Clang Checkers**

### CERT Oracle Secure Coding Standard for Java

Editors: David Svoboda, SEI/CERT

Brad Senetza, Oracle

Download the latest stable version.

***Changed***

- ERR01-J. Do not allow exceptions to expose sensitive information now contains a blurb about CVE-2015-2080.
- MSC54-J. Avoid inadvertent wrapping of loop counters had some fixes to the loop counters.

*No Java rules were added or removed.*

### CERT Secure Coding Standard for Android

Editors: Fred Long, Aberystwyth University

Lori Flynn, SEI/CERT

*No Android rules were added, removed, deprecated, or substantively changed.*

### CERT Perl Secure Coding Standard

Editor: David Svoboda, SEI/CERT

No Perl rules were added, removed, deprecated, or substantively changed.

## Our People

In the eNewsletter, we highlight the staff members behind our secure coding research. This issue we feature Aaron Ballman.

**Aaron Ballman** is a Software Security Engineer at CERT. He is an active developer on the Clang

open source C/C++/Objective-C compiler, focusing primarily on front-end development. Aaron has over a decade of experience writing commercial compilers for various programming languages, as well as developing cross-platform C and C++ frameworks.

He is the author of Ramblings on REALbasic (2009), the CERT C++ Coding Standard (Coming Soon!), and one of the authors of the CERT C Coding Standard (2014). He is currently a voting member of ISO/IEC JTC1/SC22/WG21, the C++ standards committee.

When he's not writing code, Aaron is a Women's Flat Track Roller Derby official who skates under the name Flash Drive, a director for Penobscot Community Health Care, and the caretaker of two dogs, two cats, and six chickens.

## Secure Coding Resources

**Read** Addressing the Shortfall of Secure Software Developers through Community College Education blog post.

**Watch** Secure Software Development Landscape webinar.

**Watch** From Secure Coding to Secure Software webinar.

**Subscribe to Our eNewsletter**

Join the SEI CERT Secure Coding Community



Software Engineering Institute, Carnegie Mellon University | 4500 Fifth Avenue | Pittsburgh | PA | 15213