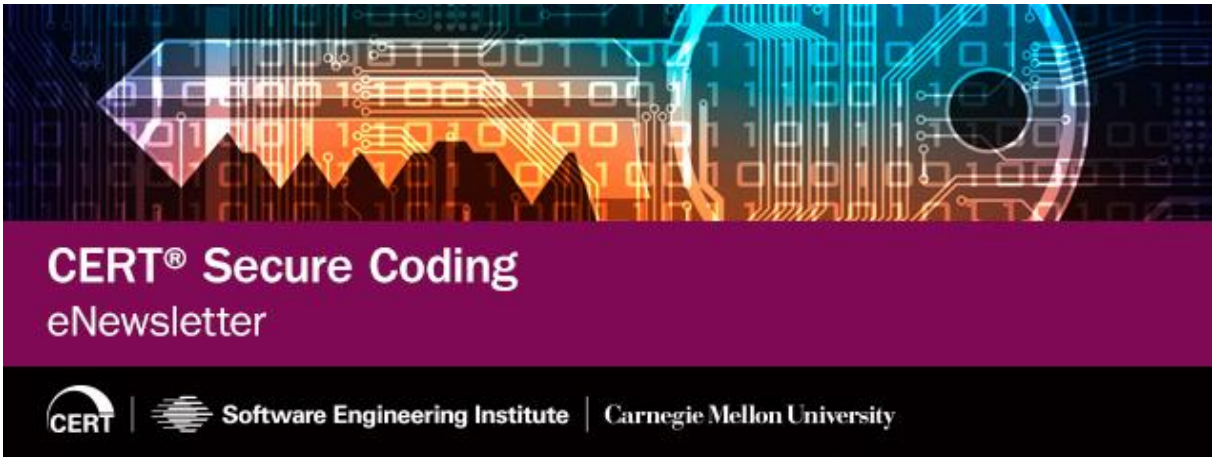
SHARE:

Join Our Email List



Fall Edition 2016

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

Another season has passed, and while we had a pleasant and mild start to the season, it's now getting cold in Pittsburgh.

In this newsletter, we highlight some recent and upcoming changes to the wiki accounts and structure, in addition to changes in the guidelines. We also highlight recent and future events that you might be interested in. Many of these events were open to the public, so we have linked to the materials in case you were unable to attend.

We hope you find this information useful. As we prepare to finish this year and get ready for the next, let us know your thoughts about our work and send us any challenges you think we should address.

Thanks,

Bob Schiela

SEI CERT Standard Publications

We have developed and published a set of errata for the SEI CERT C Coding Standard, 2016 Edition. We will continue to publish identified errata in this location.

We expect the final version of the SEI CERT C++ Coding Standard to be available in the next couple of months. We plan to publish this standard as a free PDF, similar to the release of the SEI CERT C Coding Standard, 2016 Edition.

Secure Coding Wiki Changes to Registered Accounts

There are some dormant accounts on the wiki. To more efficiently manage our site license, in early January 2017, we will disable accounts that have not been active (i.e., at least logged in) within the last 180 days. We will also disable accounts that were created more than 30 days ago if the user has not yet logged into it.

We will retain user history, contributions, and comments with attribution even after the accounts are disabled. However, users will no longer be able to log in unless they contact us to reactivate their account.

We will continue this policy moving forward and disable accounts that haven't been used in 180 days or that were created where the user did not log into it for more than 30 days.

We will also require that all registered accounts contain a valid email addresses in the user's profile. We will begin disabling accounts that do not have valid email addresses starting 1 July 2017 (following the initial 180 day window for inactive accounts).

Secure Coding Wiki Changes to Structure

Each language section now has a "Related Guidelines" summary page for each related coding standard (e.g., MISRA, MITRE CWE). These pages list the relationships that exist between CERT rules/recommendations and guidelines in external coding standards. These pages are automatically generated from the individual rule/recommendation pages.

- C
- C++
- Java

Each language section now has a "Risk Assessments" summary page that displays all rule/recommendation risk assessments for the language. These pages are automatically generated from the individual rule/recommendation pages.

- C
- C++
- Java
- Perl

We also developed guidelines for those who contribute content to the wiki, such as tool vendors that add mappings to rules.

Recent Events

Bob Schiela and the Software Engineering Institute hosted the CERT Secure Coding Symposium on 8 September 2016 in Washington, DC.

- Peter Fonash, DHS CS&C CTO, presented the keynote Strengthening the Cyber Ecosystem
- Mary Ann Davidson, Oracle CSO, presented the keynote Unleashing Your Inner Code Warrior
- David Svoboda gave a 90-minute presentation entitled Common Exploits and How to Prevent Them

David Svoboda alsogave the following presentations at JavaOne 2016 in September:

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

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

Team members made the following presentations at the SEI 2016 Research Review, which took place at the SEI in Pittsburgh on 25-26 October 2016. Links to the Secure Coding topics appear below; you can see a list of all presentations from the Research Review in the SEI's digital library.

- Experiences Developing an IBM Watson Cognitive Processing Application by Mark Sherman
- Prioritizing Alerts from Static Analysis with Classification Models by Lori Flynn
- Automated Code Repair by Will Klieber
- Establishing Coding Requirements for Non-Safety-Critical C++ Systems by David Svoboda

Lori Flynn chaired the SPLASH co-hosted workshop, Mobile! 2016, which took place on 31 October 2016 in Amsterdam, The Netherlands.

The following papers and tutorials were presented at the IEEE CyberSecurity Development (SecDev) Conference, 3-4 November 2016:

- Static Analysis Alert Audits: 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

Mark Sherman presented Experiences Developing an IBM Watson Cognitive Processing Application to Support Q&A of Application Security (Software Assurance) Diagnostics (co-authored with Lori Flynn and Chris Alberts) at the AAAI 2016 Fall Symposium on 18 November 2016.

Upcoming Events

Bob Schiela will present at the Software Assurance Community of Practice (SwA CoP) meeting in early December.

We plan to release the SEI CERT C++ Coding Standard in the next couple of months. Watch for our notices.

David Svoboda will give a Secure Coding Tutorial at the Software Solutions Symposium 2017, in Arlington, VA on March 2017, where other software engineering and security presentations and hands-on tutorials will be given. Topics include machine learning and software engineering, security engineering risk analysis, requirements elicitation, and software supply chain risk management. Registration is now open.

SEI CERT Secure Coding Standard Updates

CERT C Coding Standard

Editors: Aaron Ballman, SEI/CERT
David Svoboda, SEI/CERT

Download the latest stable version.

No C rules were added or removed.

Changed

- FIO21-C. Do not create temporary files in shared directories
  First, the severity is now Medium, as there seems to be no way to achieve privilege escalation or remote code execution through misuse of temporary files. Second, the `tmpfile()` entry now indicates that this system call can create files with reduced permissions, as indicated by POSIX.1-2008.
- FLP30-C. Do not use floating-point variables as loop counters
  Clarified the meaning of "loop counter"; corrected typo with a floating-point literal last compliant solution.
- DCL38-C. Use the correct syntax when declaring a flexible array member
  Discussed a compiler extension as though it were a standard feature; removed mention of the compiler extension.
- STR34-C. Cast characters to unsigned char before converting to larger integer sizes
  Corrected an off-by-one error with the size of the array declared for the second noncompliant & compliant code pair.
- FIO41-C. Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects
  Corrected a typo with a noncompliant code example where the code was missing a closing parenthesis.
- FIO45-C. Avoid TOCTOU race conditions while accessing files
  Corrected a typo in a compliant solution where a semicolon was missing.
- The "Expanding Buffer" solution in FIO20-C. Avoid unintentional truncation when using fgets() or fgetws() has several fixes: First, it prevents integer wrapping on the input size. Second, it avoids subtracting two null pointers. Finally, it uses `strcpy()` rather than `stract()` which would have undefined behavior when trying to concatenate to the initial buffer, as it is uninitialized.

New Clang Checkers

- MSC30-C. Do not use the rand() function for generating pseudorandom numbers

CERT C++ Secure Coding Standard

Editors: Aaron Ballman, SEI/CERT
David Svoboda, SEI/CERT

No C++ rules were added.

Changed

Due to the upcoming initial publication of the SEI CERT C++ Coding Standard 2017 Edition, several rules were renumbered to remove gaps in the rule titles for a chapter.

- INT50-CPP. Do not cast to an out-of-range enumeration value
  Reworded the first compliant solution to more clearly express the difference between it and the noncompliant code example. Also, clarified the risk assessment section.
- STR51-CPP. Do not attempt to create a std::string from a null pointer
  Clarified the title, performed minor wordsmithing on normative rule text. Was previously called STR51-CPP. Do not pass a null pointer to char_traits::length().
- EXP51-CPP. Do not delete an array through a pointer of the incorrect type
  Fixed the Risk Assessment section, removed superfluous code from the noncompliant code example/CS pair.
- EXP52-CPP. Do not rely on side effects in unevaluated operands
  Called out the fact that the first exception to this rule still requires the programmer to comply with PRE31-C. Avoid side effects in arguments to unsafe macros.
- EXP50-CPP. Do not depend on the order of evaluation for side effects
  Added an additional noncompliant & compliant code pair demonstrating the order of evaluation for overloaded operators.
- ERR58-CPP. Handle all exceptions thrown before main() begins executing
  Clarified the intent of the rule to cover exceptions being thrown that cannot be caught by `main()`. Now allowing static or thread-local objects to be declared at function block scope, since exceptions thrown from constructors of such objects can still be caught. Added a new noncompliant code example/CS pair, and a new CS to an existing noncompliant code example/CS pair. Changed the title of the rule; was previously called ERR58-CPP. Constructors of objects with static or thread storage duration must not throw exceptions.
- EXP57-CPP. Do not cast or delete pointers to incomplete classes
  Used `dynamic_cast` in the last CS, rather than `static_cast`.
- EXP60-CPP. Do not pass a nonstandard-layout type object across execution boundaries
  Clarified some explanatory text, and removed an incorrect noncompliant & compliant code pair that was better expressed in another rule.
- EXP62-CPP. Do not access the bits of an object representation that are not part of the object's value representation
  Added a noncompliant & code pair that used to live in EXP60-CPP. Do not pass a nonstandard-layout type object across execution boundaries. Also, clarified the wording of the exception to better define what access means and fixed a mistake in the introductory text regarding object representations.
- OOP57-CPP. Prefer special member functions and overloaded operators to C Standard Library functions
  Clarified a code example and added a cross reference.

- MEM51-CPP. Properly deallocate dynamically allocated resources
  Added entries to the table of paired allocation & deallocation functions for class-specific overloads of `operator new` and `operator delete` (and the array forms).
- FIO51-CPP. Close files when they are no longer needed
  Added a new compliant solution, and called out the code example's relationship to ERR50-CPP. Do not abruptly terminate the program.
- ERR50-CPP. Do not abruptly terminate the program
  Added `std::quick_exit()` to the list of function calls to avoid. Also, changed a bulleted list into a numbered list and clarified that calling `std::exit()` is acceptable.
- ERR56-CPP. Guarantee exception safety
  The compliant solution now copies the correct number of elements from the source object.
- OOP52-CPP. Do not delete a polymorphic object without a virtual destructor
  Changed the title to say "delete" rather than "destroy" and added another noncompliant code example showing that smart pointers also suffer from the same problem.
- CON54-CPP. Wrap functions that can spuriously wake up in a loop
  Corrected the predicate logic in the noncompliant code example/CS pair.
- DCL58-CPP. Do not modify the standard namespaces
  Moved from the MSC section to the DCL section; was previously called MSC53-CPP. Do not modify the standard namespaces. Added another compliant solution which demonstrates one situation where you are allowed to modify the standard namespace.
- OOP54-CPP. Gracefully handle self-copy assignment
  Removed mention about self-move assignment, making the rule only address self-copy assignment. There are questions as to whether self-move assignment truly leads to security or correctness concerns, especially given that the STL does not prohibit the behavior.
- OOP58-CPP. Copy operations must not mutate the source object
  Changed the title to more clearly state the intent of the rule; was previously called OOP58-CPP. Copy operations must mutate only the destination of the copy.
- CON51-CPP. Ensure actively held locks are released on exceptional conditions
  Added explanatory text describing some of the differences between the various locking helper classes.
- DCL60-CPP. Obey the one-definition rule
  Moved from the MSC section to the DCL section; was previously called MSC52-CPP. Obey the One-Definition Rule.
- ERR53-CPP. Do not reference base classes or class data members in a constructor or destructor function-try-block handler
  Removed the noncompliant & compliant code pair showing a function-try-block in the destructor; the noncompliant code example was far-fetched, and the lack of a conforming compliant solution made the exposition unenlightening.
- MSC54-CPP. A signal handler must be a plain old function
  Updated the last compliant solution to demonstrate a reasonable way to improve the noncompliant code example.
- ERR57-CPP. Do not leak resources when handling exceptions
  Corrected the exception handler comments in the first set of noncompliant & compliant code pairs..
- CTR55-CPP. Do not use an additive operator on an iterator if the result would overflow
  Removed a faulty noncompliant & compliant code pair.

Removed

- EXP56-CPP. Do not cast pointers into more strictly aligned pointer types
  This rule was already covered by EXP36-C. Do not cast pointers into more strictly aligned pointer types.

New Clang Checkers

- MSC50-CPP. Do not use std::rand() for generating pseudorandom numbers

CERT Oracle Secure Coding Standard for Java

Editor: David Svoboda, SEI/CERT

Download the latest stable version.

No Java rules were added or removed.

Changed

- NUM09-J. Do not use floating-point variables as loop counters
  Clarified the meaning of "loop counter"
- OBJ11-J. Be wary of letting constructors throw exceptions
  Corrected "Related Vulnerabilities" reference from CVE-2008-5339 to CVE-2008-5353.
- SER12-J. Prevent deserialization of untrusted data
  Now forbids untrusted data (as opposed to untrusted classes).
- SER02-J. Sign then seal objects before sending them outside a trust boundary
  Now mandates signing & sealing for all objects (it used to only apply to sensitive objects).
- SER13-J has been demoted to a recommendation; it is now:
  SEC58-J. Deserialization methods should not perform potentially dangerous operations
  This guideline no longer mentions whitelisting (which is addressed in SER12-J. Prevent deserialization of untrusted data). Therefore its whitelisting compliant solution has been replaced with a compliant solution which always throws an exception on deserialization, rather than perform a dangerous operation.
- ERR03-J. Restore prior object state on method failure
  The code examples have improved comments to explain the math behind the behavior of the noncompliant code example.
- MSC59-J. Limit the lifetime of sensitive data
  Now has a bugfix in its last compliant solution: The buffer is explicitly zeroed out with an additional `put()` operation, as `buffer.clear()` does not erase any sensitive information in the buffer.

CERT Secure Coding Standard for Android

Editor: 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. In this issue we feature Dr. Lori Flynn.

Dr. Lori Flynn is a software security researcher at the CERT Division of Carnegie Mellon University's Software Engineering Institute. Her research focuses on automating analysis of software for security. Prior to joining the SEI, she co-invented a patented static analysis method to create signatures for polymorphic viruses. Flynn is part of the CERT team that developed DidFail, the first static taint flow analyzer for Android app sets, and she is currently working on a research project that will increase its precision while retaining its speed. She also leads a research project working to accurately and automatically classify and prioritize alerts from code analysis tools.

Subscribe to Our eNewsletter

Join the SEI CERT Secure Coding Community