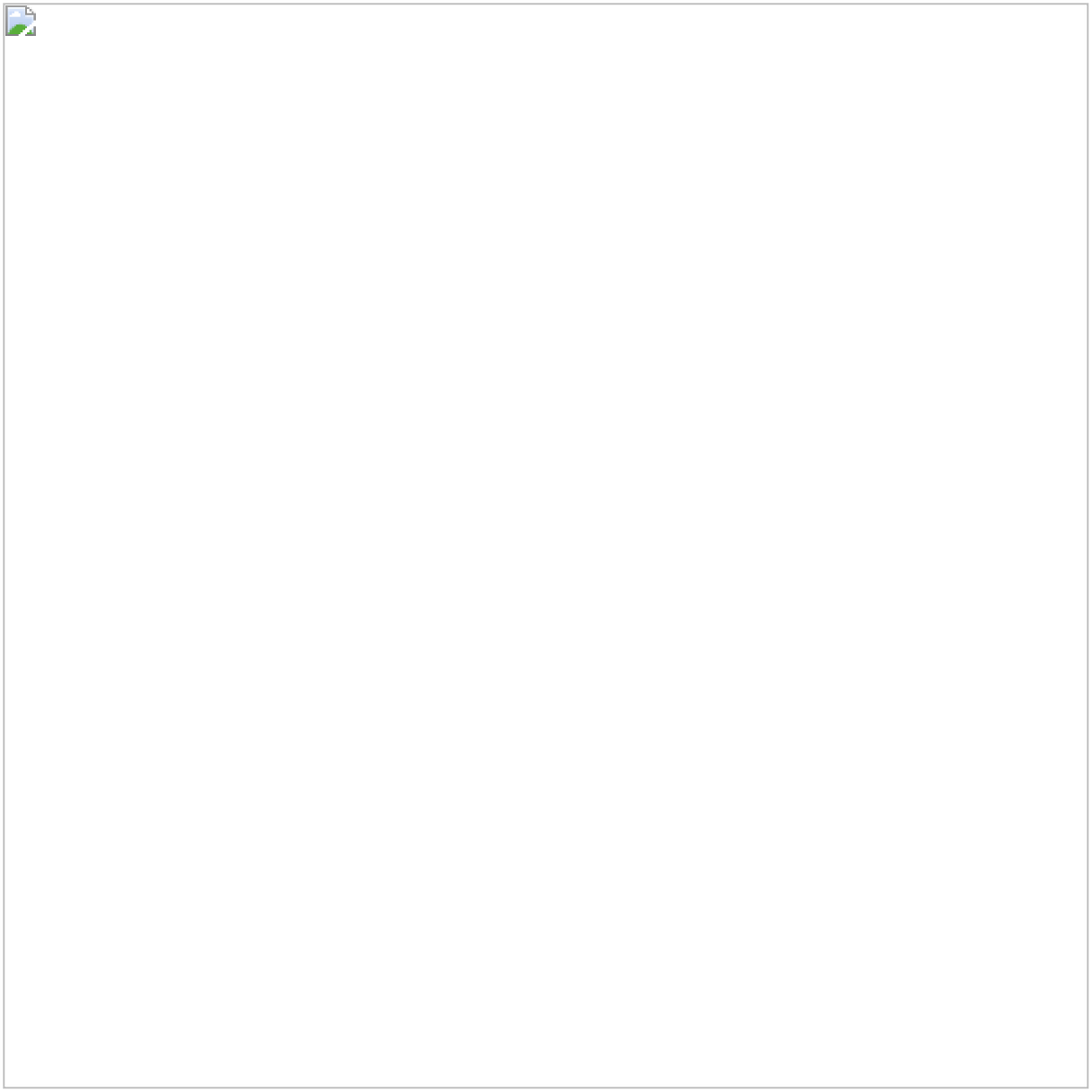


February / March 2015

News

We just finished reorganizing and reformatting the C, C++, Java, and Perl wiki spaces, and the CERT Coding Standards wiki landing page. To provide a more consistent experience for wiki reviewers, content developers, and maintainers, we made significant changes to the underlying structure of each space. These structural changes fixed longstanding problems with breadcrumbs and enabled Page Tree views of each space. You'll also see a logical order to the left sidebar navigation and simpler, more user-friendly top-level wiki pages.

We revised the numbering in Java and C++ and introduced and applied a consistent naming scheme across the wiki content. (You can still search to find the rule and recommendation numbering in C and Perl wiki spaces.) Java guidelines became Java recommendations, making it more consistent with the other wiki spaces. If you have input about the new structure and design, please contact us.

Major development work continues on the CERT C++ Coding Standard, which has a long list of added, changed, and removed rules in this newsletter.

Lori Flynn and Will Klieber published a new Technical Report on work done with a team of Carnegie Mellon University grad students (Will Snavely, Jonathan Burket, Jonathan Lim, and Wei Shen). This report describes recent significant enhancements to DidFail, the CERT static taint analyzer for sets of Android apps.

Oracle released the January 2015 Critical Patch Update on January 20, 2015. This Critical Patch Update provides 169 new fixes for security issues across a wide range of product families including Oracle Database, Oracle Fusion Middleware, Oracle Enterprise Manager, Oracle E-Business Suite, Oracle Supply Chain Suite, Oracle PeopleSoft Enterprise, Oracle JDEdwards EnterpriseOne, Oracle Siebel CRM, Oracle iLearning, Oracle Java SE, Oracle Sun Systems Products Suite, Oracle Linux and Virtualization, and Oracle MySQL.

How are you using the CERT Secure Coding Standards?

As a reader of this eNewsletter, your input is important to us. Submit your comments and let us know how you are using CERT Secure Coding Standards.

Language Standards Updates

CERT C Coding Standard

Editors: Martin Sebor and Aaron Ballman (SEI/CERT)

Changed

- FIO24-C. Do not open a file that is already open
  Downgraded from a rule (was previously labeled as FIO31-C). This rule covers implementation-defined behavior and basically relies on programmer intent to determine whether or not it is a security flaw. Also, it is mostly covered by FIO45-C. Avoid TOCTOU race conditions while accessing files when the implementation does allow the same file to be opened multiple times.
- MEM34-C. Only free memory allocated dynamically
  Added information about CVE-2015-0240, which describes a remote code execution exploit with Samba >= 3.5 where an uninitialized pointer value was passed to a resource freeing function.
- FLP34-C. Ensure that floating-point conversions are within range of the new type
  Corrected a compliant solution to use a runtime range check instead of a compile-time assertion.

CERT C++ Secure Coding Standard

Editors: Martin Sebor and Aaron Ballman (SEI/CERT)

Added

- MSC17-CPP. Do not use deprecated or obsolescent functionality
  Stubbed out this recommendation as a C++ extension to MSC24-C. Do not use deprecated or obsolescent functions, with some bare-bones content. This recommendation is incomplete.
- MSC56-CPP. A signal handler must be a Plain Old Function
- EXP54-CPP. Do not access an object outside of its lifetime
- ERR51-CPP. Handle all exceptions
- MEM55-CPP. Honor replacement dynamic storage management requirements
- OOP56-CPP. Honor replacement handler requirements
- EXP59-CPP. Use offsetof() on valid types and members

Changed

- CTR54-CPP. Do not subtract iterators that do not refer to the same container
  Effectively rewritten; was previously called CTR36-CPP. Do not subtract or compare two pointers or iterators that do not refer to the same array or container
- STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
  Added a new NCCE/CS pair to cover unformatted stream input functions `read()` and `readsome()`, which do not null-terminate their buffers.
- FIO20-CPP. Do not rely on an ending null character when using read()
  Downgraded from a rule (was previously labeled as FIO45-CPP), and corrected some wording that incorrectly suggested `read()` would sometimes null-terminate the given buffer. This matches the analogous C recommendation FIO17-C. Do not rely on an ending null character when using fread().
- FIO21-CPP. Do not simultaneously open the same file multiple times
  Downgraded from a rule (was previously labeled as FIO31-CPP). This rule covers implementation-defined behavior and basically relies on programmer intent to determine whether it is a security flaw.
- FIO51-CPP. Close files when they are no longer needed
  Rewrote the contents of the rule and changed the title to be more consistent with the analogous C rule. This rule was previously called FIO42-CPP. Ensure files are properly closed when they are no longer needed.
- FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
  Rewrote the contents of the rule and changed the title to specify file streams explicitly, removing the mention of flushing. This rule was previously called FIO39-CPP. Do not alternately input and output from a stream without an intervening flush or positioning call.
- ERR52-CPP. Do not use setjmp() or longjmp()
  Added an explicit mention of invoking undefined behavior to the normative text and removed the exception allowing `setjmp` and `longjmp` under tightly controlled circumstances.
- ERR58-CPP. Constructors of objects with static or thread storage duration must not throw exceptions
  Substantive changes to the content of the rule. Was previously called ERR41-CPP. Constructors of objects with static storage duration must not throw exceptions.
- INT50-CPP. Do not cast to an out-of-range enumeration value
  Updated the normative wording to clarify that unscoped enumerations can be fixed; added another compliant solution to demonstrate this fact.

- MEM03-CPP. Clear sensitive information stored in returned reusable resources
  Minor updates to the code examples but no functional changes to the wording of the recommendation.
- MSC52-CPP. Obey the One Definition Rule
  Rewrote the contents of the rule and clarified the way the ODR can be violated.
- OOP10-CPP. Do not assume that copy constructor invocations will not be elided
  Downgraded from a rule (was previously labeled as OOP36-CPP). This rule covers implementation-defined behavior and basically relies on programmer intent to determine whether it is a security flaw.
- CTR51-CPP. Use valid references, pointers, and iterators to reference elements of a container
  Completed the table of container functions that invalidate iterators, references, or pointers.
- OOP50-CPP. Do not invoke virtual functions from constructors or destructors
  Rewrote the contents of the rule and added a second exception.
- ERR50-CPP. Do not call std::terminate(), std::abort(), or std::_Exit()
  Rewrote the contents of the rule and changed the title to forbid calls to `std::abort()`, but provided an exception for critical errors. Was previously called ERR30-CPP. Try to recover gracefully from unexpected errors.
- EXP50-CPP. Do not depend on the order of evaluation for side effects
  Rewrote the contents of the rule, but the NCCE/CS pairs remain roughly identical to their previous form. Changed the title to be more consistent with the analogous C rule. Was previously called EXP30-CPP. Do not depend on order of evaluation between sequence points.
- DCL55-CPP. Overloaded postfix increment and decrement operators should return a const object
  Corrected a think-o with the compliant solution.
- MEM50-CPP. Do not access freed memory
  Added an NCCE/CS pair to handle indirection through the result of a zero-sized allocation.
- Content from the CERT C Coding Standard that also applies in C++ is now explicitly called out in each of the section indexes.
- MSC51-CPP. Ensure your random number generator is properly seeded
  Improved the definition of what a proper seed value is, added another NCCE, added a related vulnerability.

Removed

- FIO30-CPP. Exclude user input from format strings
  This rule is entirely covered by FIO30-C. Exclude user input from format strings.
- MEM00-CPP. Don't use auto_ptr where copy semantics might be expected
  Since `auto_ptr` is deprecated, this recommendation is already covered by the stub recommendation MSC17-CPP. Do not use deprecated or obsolescent functionality.
- SIG30-CPP. Call only asynchronous-safe functions within signal handlers
  This rule is covered by SIG30-C. Call only asynchronous-safe functions within signal handlers.
- SIG31-CPP. Do not access shared objects in signal handlers
  This rule is covered by SIG31-C. Do not access shared objects in signal handlers.
- CON30-CPP. Ensure all threads exit before exiting main
  This rule applies to POSIX, where `pthread_exit()` exists, but it does not apply to C++.
- DCL30-CPP. Declare objects with appropriate storage durations
  This rule was rewritten to be EXP54-CPP. Do not access an object outside of its lifetime.

- The PRE, FLP, ENV, and SIG sections have been removed due to a lack of expected content for them in C++. Rules related to the preprocessor, floating-point types, the environment, or signals will exist in the Rule 49: Miscellaneous (MSC) section instead.
- The SonarQube Plugin and G++ analyzer pages have been removed. The former appears unused in any Automated Detection section, and the latter was consolidated with the GCC analyzer page.

CERT Oracle Secure Coding Standard for Java
Editors: Adam O'Brien (Oracle) and David Svoboda (SEI/CERT)

The Java Coding Guidelines are now publicly available.

Changed

- NUM00-J. Detect or prevent integer overflow now has a compliant solution describing the Math.`*Exact()` methods, which can be used to detect overflow when adding, subtracting, or multiplying integer values.
- Fixed a bug in the first compliant solution in MSC59-J. Limit the lifetime of sensitive data; it now safely clears its password even when an exception is thrown.

CERT Secure Coding Standard for Android
Editors: Fred Long (Aberystwyth University) and 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.

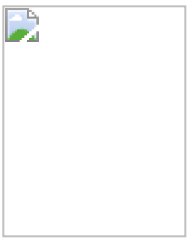
Upcoming Events

Webinar:
Using DidFail to Analyze Flow of Sensitive Information in Sets of Android Apps
by Will Klieber and Lori Flynn. May 7, 2015
Details at https://www.webcaster4.com/Webcast/Page/139/7678

RSAConference | Where the world talks security
April 20-24, 2015 San Francisco, CA.
http://www.rsaconference.com/events/us15

Our People
In the eNewsletter, we highlight staff members behind our secure coding research. This month we feature Dr. Lori Flynn and Dr. Will Klieber.

Dr. Lori Flynn is a researcher in the Secure Coding Initiative of the CERT division of Carnegie



Mellon University's Software Engineering Institute. Her work includes research and development of new static analyses and secure coding rules. Flynn is part of the team that developed DidFail, the first static taint flow analyzer for Android app sets, and she is continuing work to enhance it. Flynn also researches static analysis methods for detecting Java malware.

Dr. Will Klieber is a researcher at the CERT division of the Software Engineering Institute and an author of DidFail. His work has focused on static analysis of Android apps and detection of potentially malicious Java source code. Prior to joining the CERT division, Klieber was a doctoral student at Carnegie Mellon University's Computer Science Department, where he worked in the area of Quantified Boolean Formulas (QBF) and its application to verification of hardware and software.

Secure Coding Resources

Read the Technical Note Making DidFail Succeed: Enhancing the CERT Static Taint Analyzer for Android App Sets by Jonathan Burket, Lori Flynn, Will Klieber, Jonathan Lim, Wei Shen and William Snavely

Watch the Webinar Advancing Cyber Intelligence Practices Through the SEI's Consortium by Jay McAllister and Melissa Ludwick

Read the Blog post Secure Coding to Prevent Vulnerabilities

Subscribe to Our eNewsletter

Join the SEI CERT Secure Coding Community