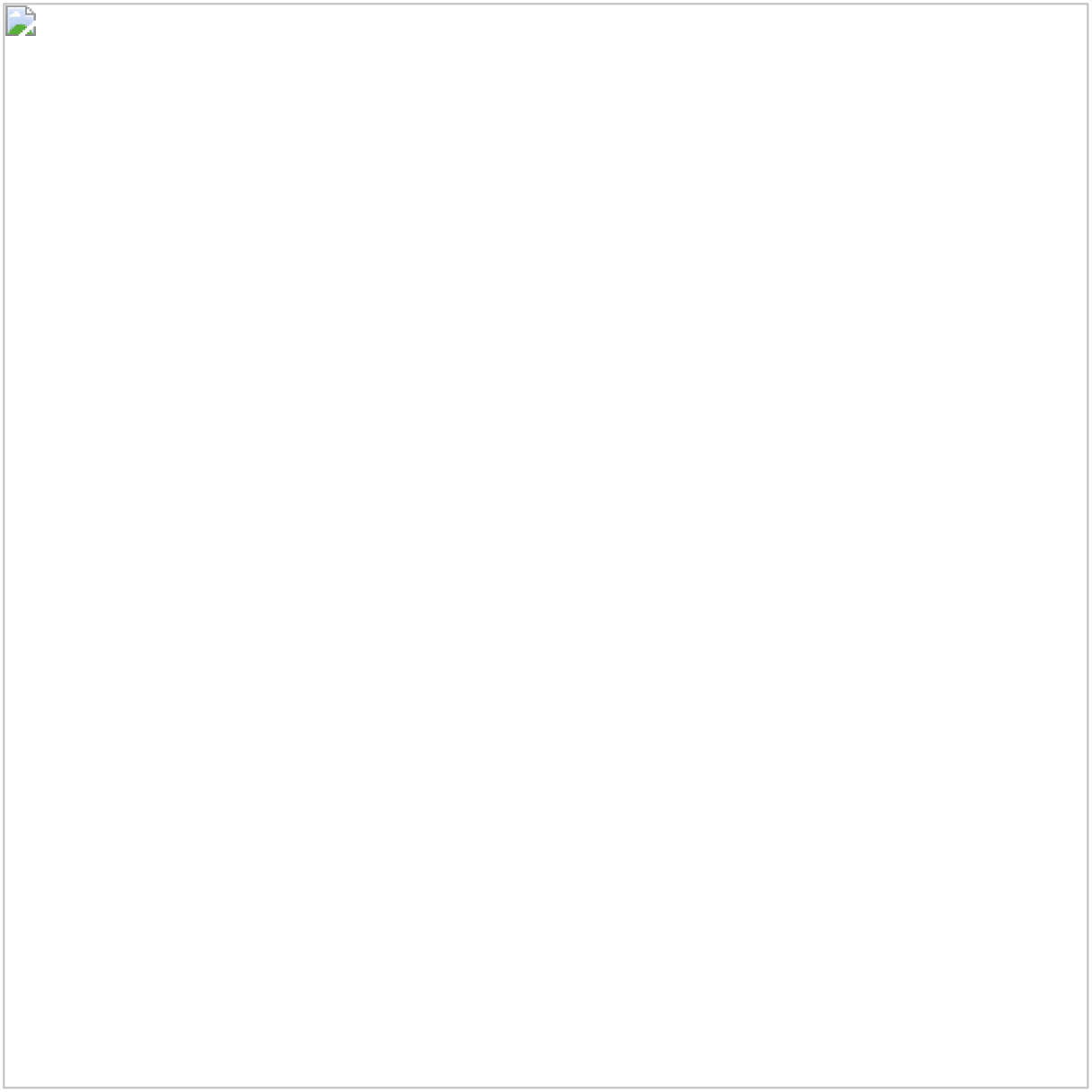

December / January 2015

News

Major development work continues on the CERT C++ Coding Standard, which has a long list of added, changed, and removed rules in this newsletter. We are also reorganizing both the Java Coding Guidelines and the The CERT Oracle Coding Standard for Java to make them easier to navigate. Part of this effort is to adopt the unique identifiers from the coding standard (see Guidelines) to the secure coding guidelines. We are updating all the Java rules and guidelines for Java Standard Edition (SE) 8, and we encourage the community to participate in this project. As part of this change, we have created a new section in the Java rules for rules involving characters and strings: 04. Characters and Strings (STR).

Lori Flynn and Will Klieber led a team of Carnegie Mellon University grad students (Will Snavely, Jonathan Burket, Jonathan Lim, and Wei Shen) on a semester-long project that significantly enhanced DidFail, our static taint flow analyzer for sets of Android apps. First, the team developed a new framework for testing the DidFail analyzer, which includes a setup for cloud-based testing and instrumentation to measure performance of the analyzer. The new setup for cloud-based testing enables us to take advantage of Amazon's powerful virtual machines and to use virtual machines in parallel for faster test completion. Second, DidFail was modified to use the most current version of FlowDroid and Soot, and the new version of DidFail was able to successfully process three times as many apps as it was able to previously, from a set of 90 apps randomly chosen from a large collection. Third, initial enhancements were made to DidFail, which moved us closer to the goal of analyzing all types of components and shared static fields. Fourth, the team developed new test apps, which test the analytical features added to DidFail. Finally, testing was done, using this improved DidFail analyzer and the cloud-based testing framework, on the new test apps and also on apps from the Google Play store. The grad students did excellent work, and Lori and Will are currently working with them to write an SEI technical report that will detail the testing framework, enhancements to DidFail, newly developed test apps, and test results. The new code developed for this project will be published soon.

How are you using the CERT Secure Coding Standards?

As a reader of this eNewsletter, we want to hear from you. 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)

Added

- Added an analyzer section for Clang.

Changed

- EXP44-C. Do not rely on side effects in operands to sizeof, _Alignof, or _Generic
  Updated examples and text regarding the `_Alignof` operator because its operand is only a type (not an arbitrary expression). Added an exception to the rule for idiomatic reads of a volatile-qualified object.
- DCL37-C. Do not declare or define a reserved identifier
  Added an exception to the rule for idempotent macro definitions of reserved identifiers; these can be machine generated by tools such as autoconf and are harmless. Also added an exception for standard library implementers and compiler vendors.

CERT C++ Secure Coding Standard

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

Added

- CTR05-CPP. Use explicit cv- and ref-qualifiers on auto declarations in range-based for loops
- EXP32-CPP. Do not rely on side effects in unevaluated operands
- DCL40-CPP. Destructors and deallocation functions must be declared noexcept
- MEM31-CPP. Properly deallocate dynamically allocated resources
- MEM33-CPP. Explicitly initiate and terminate object lifetime when performing manual lifetime management

Changed

- The definition of undefined behavior was updated to also cover cases in which the standard defines a construct to be ill-formed with no diagnostic required. This update brings our definition more in line with [into.compliance]p2, which states in part: "If a program contains a violation of a rule for which no diagnostic is required, this International Standard places no requirement on implementations with respect to that program." (This is roughly an identical definition to undefined behavior in [defns.undefined].)
- OOP39-CPP. Do not use pointer-to-member operators to access nonexistent members
  Substantive changes to the normative wording.
- DCL35-CPP. Overload allocation and deallocation functions as a pair in the same scope
  Substantive changes to the normative wording, added code examples. Was previously called "DCL35-CPP. Do not overload operator new in a different scope than operator delete."
- MEM03-CPP. Clear sensitive information stored in returned reusable resources
  Converted a CS involving `std::string::clear()` into an NCCE; `the clear()` function makes no guarantees regarding the values of the memory being cleared.
- MSC30-CPP. Do not use std::rand() for generating pseudorandom numbers
  Substantive changes to the code examples; minor updates to the normative wording. Was previously called "MSC30-CPP. Do not use the rand() function for generating pseudorandom numbers."
- MSC32-CPP. Ensure your random number generator is properly seeded
  Substantive changes to the code examples; updates to the normative wording to match that of the corresponding C rule.
- DCL32-CPP. Do not declare or define a reserved identifier
  Substantive changes to the normative wording and examples. Added an exception to the rule for idempotent macro definitions of reserved identifiers; these can be machine generated by tools such as autoconf and are harmless. Also added an exception for standard library implementers and compiler vendors.
- EXP33-CPP. Do not read uninitialized memory
  Substantive changes to the content of the rule; was previously called "EXP33-CPP. Do not reference uninitialized memory."
- CTR44-CPP. Predicate function objects should have pure call semantics
- Substantive changes to the content of the rule; was previously called "CTR44-CPP. Predicate functors should not have non-static non-const data fields."

Removed

- ENV00-CPP. Do not store the pointer to the string returned by getenv()
  This recommendation is covered entirely by ENV00-C. Do not store objects that can be overwritten by multiple calls to getenv() and similar functions.
- ENV01-CPP. Do not make assumptions about the size of an environment variable
  This recommendation is covered entirely by ENV01-C. Do not make assumptions about the size of an environment variable.

- SIG33-CPP. Do not recursively invoke the raise() function
  This rule is covered entirely by SIG30-C. Call only asynchronous-safe functions within signal handlers.
- SIG34-CPP. Do not call signal() from within interruptible signal handlers
  This rule is covered entirely by SIG34-C. Do not call signal() from within interruptible signal handlers.
- FLP37-CPP. Check floating point inputs for exceptional values
  This rule was a duplicate of FLP04-CPP. Check floating point inputs for exceptional values; a recommendation is more appropriate than a rule in this instance.
- ERR38-CPP. Deallocation functions must not throw exceptions and ERR33-CPP. Destructors must not throw exceptions
  These rules were replaced by DCL40-CPP. Destructors and deallocation functions must be declared noexcept.
- MEM41-CPP. Declare a copy constructor, a copy assignment operator, and a destructor in a class that manages resources
  The parts of this rule that are normatively enforceable are covered by MEM31-CPP. Properly deallocate dynamically allocated resources. The remainder of the rule may be resurrected as a recommendation in the future, focusing on the Rule of Three (Five).
- MEM34-CPP. Only free memory allocated dynamically and MEM39-CPP. Resources allocated by memory allocation functions must be released using the corresponding memory deallocation function
  These rules were combined and superseded by MEM31-CPP. Properly deallocate dynamically allocated resources.
- DCL36-CPP. Do not declare an identifier with conflicting linkage classifications
  This rule does not apply in C++ because the same entity cannot be declared with conflicting linkage classifications in a conforming implementation.

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.

Added

- Added a new section on characters and strings: 04. Characters and Strings (STR)

  These former stub rules have now been completed with compliant and noncompliant code examples:
- FIO15-J. Do not reset a servlet's output stream after committing it
- IDS14-J. Do not trust the contents of hidden form fields
- MSC11-J. Do not let session information leak within a servlet

  This new guideline has been added:
- STR50-J. Use the appropriate method for counting characters in a string

Changed

- STR01-J. Do not assume that a Java char fully represents a Unicode code points
  Now has a section describing combined characters. Also, the broken Hornig link has been replaced by a link to the Java Tutorial.
- FIO11-J. Do not attempt to read raw binary data as character data has merged with IDS12-J. Perform lossless conversion of String data between differing character

encodings to produce the new rule STR05-J. Use the charset encoder and decoder classes when more control over the encoding process is required.
- The last compliant solution in LCK10-J. Use a correct form of the double-checked locking idiom was not completely thread-safe. It has been changed to a noncompliant code example, and a compliant solution was added in to address thread safety.
- MSC00-J. Use SSLSocket rather than Socket for secure data exchange
  Now contains information about hostname validation
- IDS17-J. Prevent XML External Entity Attacks
  Severity was changed to medium.
- FIO06-J. Do not create multiple buffered wrappers on a single byte or character stream
  Now applies to both byte streams and character streams, for both input and output.
- FIO08-J. Use an int to capture the return value of methods that read a character or byte
  Rule was retitled, and the description was rewritten.

Deprecation Candidates

- FIO17-J. Pass a character set to any method converting bytes to strings

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.

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

Aaron Ballman has over a decade of experience writing commercial compilers for various languages, and is a Security Software Engineer for CERT. He is an active developer on the clang open source C/C++/Objective-C compiler.

When he's not writing code, Aaron also enjoys being outside, fishing, and reading a good book in his hammock.

Secure Coding Resources

Read the Blog Post - *Java Zero Day Vulnerabilities* by David Svoboda

Read the Paper - *C/C++ Thread Safety Analysis* by DeLesley Hutchins (Google, Inc.), Aaron Ballman and Dean F. Sutherland

Read the Technical Note - *Performance of Compiler-Assisted Memory Safety Checking* by David

Keaton and Robert C. Seacord

Watch the Webinar - *Lessons in External Dependency and Supply Chain Risk Management* by John Haller and Matthew J. Butkovic

Read the Blog Post - *Open Architectures in the Defense Intelligence Community* by Derrick H. Karimi

Subscribe to Our eNewsletter

Join the SEI CERT Secure Coding Community