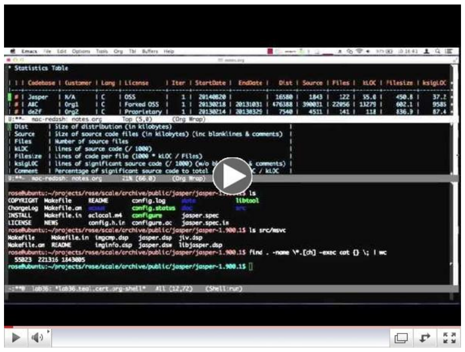
October / November 2014

News

In case you missed it, the Java Coding Guidelines are now available free online. We are making some efforts to better allocate guidelines between The CERT Oracle Secure Coding Standard for Java and the Java Coding Guidelines. We are hoping to update both The CERT Oracle Secure Coding Standard for Java and the Java Coding Guidelines to Java Standard Edition (SE) 8 and we encourage the community to participate in this project.

Major development work continues on the *CERT C++ Secure Coding Standard*, which has a long list of added, changed, and removed rules in this newsletter. Because of the close relationship between C and C++, we have also made some changes to *The CERT C Secure Coding Standard*, including adding a new floating point rule.



```
* Statistics Table

| ! | Codebase | Customer | Lang | License     | Iter | StartDate | EndDate  |   Dist | Source | Files |  kLOC | Filesize | ksigLOC
|---+----------+----------+------+-------------+------+-----------+----------+--------+--------+-------+-------+----------+--------
| # | Jasper   | N/A      | C    | OSS         |    1 |  20140820 |          |  16580 |   1843 |   122 |  55.0 |    450.8 |    37.2
| # | ABC      | Org1     | C    | Forked OSS  |    1 |  20130218 | 20131031 | 476388 | 390031 | 22056 | 13279 |    602.1 |    9585
| # | de2f     | Org2     | C    | Proprietary |    1 |  20130214 | 20130329 |   7540 |   4511 |   141 |   118 |    836.9 |    87.4
U:**-  mac-redosh: notes.org      Top (5,0)     (Org Wrap)
| Dist      | Size of distribution (in kilobytes)                         |
| Source    | Size of source code files (in kilobytes) (inc blanklines & comments) |
| Files     | Number of source files                                      |
| kLOC      | lines of source code (/ 1000)                               |
| Filesize  | Lines of code per file (1000 * kLOC / Files)                |
| ksigLOC   | lines of significant source code (/ 1000) (w/o b      & comments) |
| Comment   | Percentage of significant source code to total          C / kLOC |
U:**-  mac-redosh: notes.org      21% (66,0)    (Org Wrap)

rose@ubuntu:~/projects/rose/scale/archive/public/jasper/jasper-1.      $ ls
COPYRIGHT  Makefile     README      config.log     auto        libtool
ChangeLog  Makefile.am  acaux       config.status  doc         src
INSTALL    Makefile.in  aclocal.m4  configure      jasper.spec
LICENSE    NEWS         config.h.in configure.ac   jasper.spec.in
rose@ubuntu:~/projects/rose/scale/archive/public/jasper/jasper-1.900.1$ ls src/msvc
Makefile     Makefile.in  imgcmp.dsp   jasper.dsp  jiv.dsp
Makefile.am  README       imginfo.dsp  jasper.dsw  libjasper.dsp
rose@ubuntu:~/projects/rose/scale/archive/public/jasper/jasper-1.900.1$ find . -name \*.[ch] -exec cat {} \; | wc
  55823  221316 1843005
rose@ubuntu:~/projects/rose/scale/archive/public/jasper/jasper-1.900.1$ 

-:**@  lab36: *lab36.teal.cert.org-shell*   All (12,72)    (Shell:run)
```

Source Code Analysis Laboratory (SCALe) Demo

The CERT Division has produced a set of SCALe Demonstration Videos narrated by David Svoboda. These videos illustrate the process of auditing a small C codebase using our Source Code Analysis Laboratory (SCALe)..

We continue to improve DidFail, our static taint flow analyzer for Android app sets. Instructions for installing the latest version of DidFail are here, and more work is in progress to further improve its soundness and precision.

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 (Cisco Systems, Inc.) and Aaron Ballman (SEI/CERT)

Added

- FLP37-C. Do not use object representations to compare floating-point values

Changed

- INT30-C. Ensure that unsigned integer operations do not wrap includes a brief description of CVE-2014-4377.
- MSC12-C. Detect and remove code that has no effect or is never executed has been merged with MSC07-C. Detect and remove dead code.

Depreciated

- MSC07-C. Detect and remove dead code
- STR08-C. Use managed strings for development of new string manipulation code

CERT C++ Secure Coding Standard

Editors: Martin Sebor (Cisco Systems, Inc.) and Aaron Ballman (SEI/CERT)

The C++ Secure Coding Standard has been substantially restructured. Recommendations and Rules have been split into their own parent sections, which are now linked to from the main C++ page. Additionally, the Rules Versus Recommendations page has been updated to better align with the C page of the same title.

Added

- STR36-CPP. Do not pass a null pointer to char_traits::length
- EXP31-CPP. Do not delete an array through a pointer of the incorrect type

Changed

- OOP38-CPP. Gracefully handle self-assignment
  Updated the compliant solution to provide a basic exception guarantee; updated surrounding text to call this out explicitly.
- FIO18-CPP. Never expect write() to terminate the writing process at a null character
  Fixed a small think-o in the compliant solution.
- STR35-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
  Significant updates to the content; was previously called STR35-CPP. Do not copy data from an unbounded source to a fixed-length array.
- STR38-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
  Significant updates to the content; was previously called STR38-CPP. Use valid references, pointers, and iterators to reference string objects.
- STR39-CPP. Range check element access
  Significant updates to the content.
- CTR32-CPP. Use valid references, pointers, and iterators to reference elements of a container
  Significant updates to the content; was previously called CTR32-CPP. Do not use iterators invalidated by container modification.
- CTR30-CPP. Guarantee that container indices and iterators are within the valid range
  Significant updates to the content; was previously called CTR30-CPP. Guarantee that array and vector indices are within the valid range.
- CTR33-CPP. Guarantee that library functions do not form invalid iterators
  Significant updates to the content; was previously called CTR33-CPP. Guarantee that copies are made into storage of sufficient size.
- CTR39-CPP. Do not use pointer arithmetic on polymorphic objects
  Significant updates to the content; was previously called CTR39-CPP. Do not treat arrays polymorphically.
- CTR40-CPP. Provide a valid ordering predicate
  Significant updates to the content; was previously called CTR40-CPP. Use a valid ordering rule.
- CTR34-CPP. Use valid iterator ranges
  Significant updates to the content; merged in content from CTR43-CPP.
- CTR38-CPP. Do not use an additive operator on an iterator if the result would overflow
  Significant updates to the content; was previously called CTR38-CPP. Do not add or subtract an integer to a pointer or iterator if the resulting value does not refer to a valid element in the array or container.
- DCL38-CPP. Do not recursively reenter a function during the initialization of one of its static objects
  Minor modifications to normative wording, completely new NCCE/CS pair.

- EXP39-CPP. Do not cast or delete pointers to incomplete classes
  Significant updates to the content.
- EXP36-CPP. Do not cast pointers into more strictly aligned pointer types
  Significant updates to the content; was previously called EXP36-CPP. Do not convert pointers into more strictly aligned pointer types.
- EXP35-CPP. Do not access a cv-qualified object through a cv-unqualified type
  Significant updates to the content; was previously called EXP35-CPP. Do not cast away a const qualification.
- MEM45-CPP. Provide placement new with properly aligned pointers to sufficient storage capacity
  Significant updates to the content; was previously called MEM45-CPP. Provide properly aligned pointers to placement new.
- OOP37-CPP. Write constructor member initializers in the canonical order
  Significant updates to the content; was previously called OOP37-CPP. Constructor initializers should be ordered correctly.
- OOP08-CPP. Do not return references to private data
  Downgraded to recommendation; was previously called OOP35-CPP. Do not return references to private data.
- OOP34-CPP. Do not delete a polymorphic object without a virtual destructor
  Significant updates to the content; was previously called OOP34-CPP. Ensure the proper destructor is called for polymorphic objects.
- OOP33-CPP. Do not slice derived objects
  Significant updates to the content; was previously called OOP33-CPP. Do not slice polymorphic objects.
- OOP09-CPP. Ensure that single-argument constructors are marked "explicit"
  Downgraded to recommendation; was previously called OOP32-CPP. Ensure that single-argument constructors are marked "explicit."
- DCL34-CPP. Do not write syntactically ambiguous declarations
  Significant updates to the content; was previously called OOP31-CPP. Ensure object construction invocations isn't mistaken for a function variable declaration.
- MSC34-CPP. Do not modify the standard namespaces
  Significant updates to the content; was previously called MSC34-CPP. Do not modify the standard namespace.
- CTR36-CPP. Do not subtract or compare two pointers or iterators that do not refer to the same array or container
  Fixed a minor typo.
- FIO45-CPP. Do not rely on an ending null character when using read()
  Fixed a minor typo.
- ERR34-CPP. Do not use setjmp() or longjmp()
  Significant updates to the content; was previously called ERR34-CPP. Do not use longjmp.
- ERR35-CPP. Do not reference base classes or class data members in a constructor or destructor function-try-block handler
  Significant updates to the content; was previously called ERR35-CPP. A handler in a constructor or destructor's function-try-block should not reference class data.
- CTR04-CPP. Assume responsibility for cleaning up data referenced by a container of pointers
  Fixed the code examples to be vaguely more C++-like; removed a compliant solution that did not address the guideline.
- ERR36-CPP. Catch handlers should order their parameter types from most derived to least derived
  Updates to the content; was previously called ERR36-CPP. Multiple catch handlers to a try block should order their exceptions from most derived to most basic

- DCL31-CPP. Do not define a C-style variafic function
  Significant updates to the content; was previously called DCL31-CPP. Do not define variadic functions

Removed

- CTR37-CPP. Do not add or subtract an integer to a pointer to a non-array object
  This rule was covered entirely by the C rule ARR37-C. Do not add or subtract an integer to a pointer to a non-array object.
- CTR42-CPP. Never modify a set or multiset key in-place
  This rule was not required because it is not an issue for compilers conforming to C++2003 or later.
- CTR41-CPP. A container's allocator should never have a data field that is not static
  Whereas C++03 and earlier prohibited stateful allocators, C++11 and later allow for them.
- CTR43-CPP. Do not access collapsed elements from a remove(), remove_if() or unique() operation
  This rule was merged into CTR34-CPP. Use valid iterator ranges
- CTR35-CPP. Do not allow loops to iterate beyond the end of an array or container
  This rule is covered by CTR30-CPP. Guarantee that container indices and iterators are within the valid range and CTR34-CPP. Use valid iterator ranges.
- EXP32-CPP. Do not access a volatile object through a non-volatile reference
  This rule is entirely covered by the C rule EXP32-C. Do not access a volatile object through a nonvolatile reference, and the C++ rule EXP35-CPP. Do not access a cv-qualified object through a cv-unqualified type.
- PRE30-CPP. Do not create a universal character name through concatenation
  This rule is entirely covered by the C rule PRE30-C. Do not create a universal character name through concatenation.

- MSC35-CPP. Do not use goto statement to take control inside the try and catch blocks
  This rule was not required because it is not an issue for conforming compilers.
- FLP35-CPP. Take granularity into account when comparing floating point values
  This rule is entirely covered by the C recommendation FLP00-C. Understand the limitations of floating-point numbers.
- ERR31-CPP. Don't redefine errno
  This rule is entirely covered by the C rules DCL37-C. Do not declare or define a reserved identifier and MSC38-C. Do not treat a predefined identifier as an object if it might only be implemented as a macro.
- PRE10-CPP. Wrap multistatement macros in a do-while loop
  This rule is entirely covered by the C recommendation PRE10-C. Wrap multistatement macros in a do-while loop.

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

Stubs were added for the following rules:

- MSC09-J. For OAuth, ensure (a) [relying party receiving user's ID in last step] is same as (b) [relying party the access token was granted to]

- MSC10-J. Do not use OAuth 2.0 implicit grant (unmodified) for authentication

Changed

- DCL02-J. Do not modify the collection's elements during an enhanced for statement has been slightly restricted. It no longer requires explicitly declaring such variables final as long as they are not changed. The code examples have also been revamped.
- EXP01-J. Never dereference null pointers now has a compliant solution illustrating Java 8's new `Optional<>` class.
- IDS01-J. Normalize strings before validating them has a revamped intro, to make the guideline clearer.
- IDS00-J used to cover all cases of unsanitized input crossing a trust boundary. This rule has been replaced with rules decrying more specific vulnerable practices:
    - IDS00-J. Prevent SQL Injection
    - IDS16-J. Prevent XML Injection
    - IDS17-J. Prevent XML External Entity Attacks

- IDS02-J. Canonicalize path names before validating them has been moved to FIO16-J. Canonicalize path names before validating them.
- The compliant solution in IDS04-J. Safely extract files from ZipInputStream now handles subdirectories in zip files.
- IDS05-J. Use a safe subset of ASCII for file and path names has been moved to the Java Guidelines because its warnings are covered by several other rules.

Deprecation Candidates

- EXP05-J. Do not follow a write by a subsequent write or read of the same object within an expression

Removed

- IDS05-J. Use a safe subset of ASCII for file and path names is being moved to a FIO guideline

CERT Secure Coding Standard for Android

Editors: Fred Long, (Aberystwyth University) and Lori Flynn (SEI/CERT)

Added

Stubs were added for the following rules

- DRD24-J. Do not bundle OAuth security-related protocol logic or sensitive data into a relying party's app
- DRD25-J. To request user permission for OAuth, identify relying party and its permissions scope
- DRD26-J. For OAuth, use a secure Android method to deliver access tokens

CERT Perl Secure Coding Standard

Editor: David Svoboda (SEI/CERT)

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

Upcoming Events and Training

Presentation
David Svoboda will present "Another Java 0-Day Exploit" to the Steel City Information Security group, on November 13, 2014.

Our People

In the eNewsletter, we highlight staff members behind our secure coding research. This month we feature Fred Long.

Fred Long is a senior lecturer and director of learning and teaching in the Department of Computer Science, Aberystwyth University, United Kingdom. He lectures on formal methods; Java, C++, and C programming paradigms and programming-related security issues. He is chairman of the British Computer Society's Mid-Wales Sub-Branch. Fred has been a Visiting Scientist at the Software Engineering Institute since 1992. Recently, his research has involved the investigation of vulnerabilities in Java.

Secure Coding Resources

Read the blog post *Thread Safety Analysis in C and C++* by Aaron Ballman

Read the book *Java Coding Guidelines* by the CERT Secure Coding Team

Watch the SEI Panel Discussion - *Heartbleed: Analysis, Thoughts, and Actions*

Subscribe to Our eNewsletter

Join the SEI CERT Secure Coding Community

Constant Contact