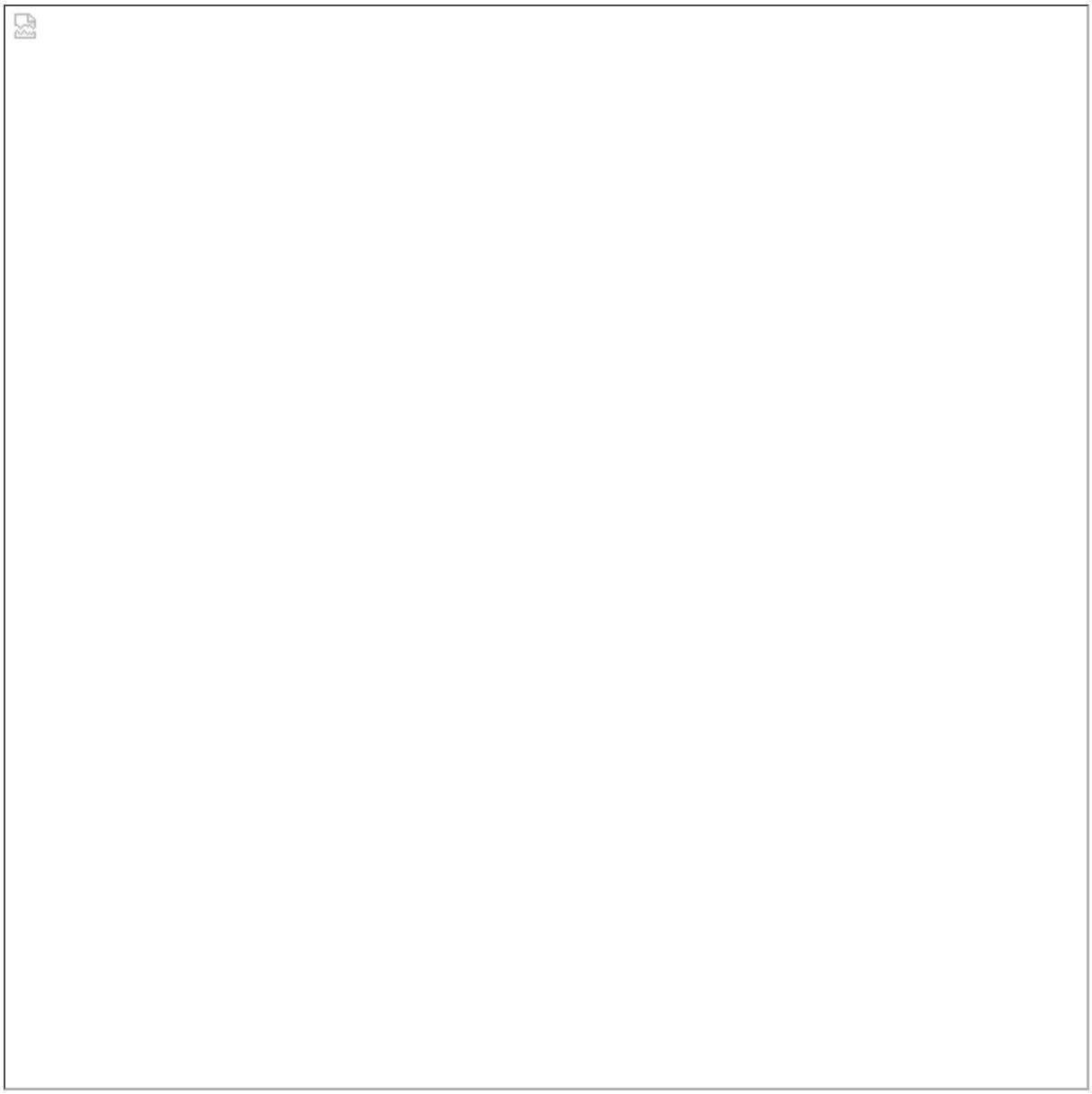

August / September 2014

News

Robert Seacord is presenting at the TB3335-Why are we still not programming securely? at the HP Protect 2014 conference that will be held September 8-11 at the Washington Hilton in Washington, D.C. Good seats are still available.

Aaron Ballman has begun his update to the CERT C++ Secure Coding Standard, which is being formulated on the C++ Coding Standard Development Guidelines page. Please feel free to join

the discussion as we plan this major update.

Automated detection mappings have been updated, or are underway, for several analyzers, including PRQA QA-C, Coverity Prevent, GrammaTech CodeSonar, and SonarQube Plugin.

The SEI report titled Performance of Compiler-Assisted Memory Safety Checking, authored by David Keaton and Robert Seacord, has been published on the SEI website. This technical note describes the criteria for deploying a compiler-based memory safety checking tool and the performance that can be achieved with two such tools whose source code is freely available.

David Svoboda and Robert Seacord will be presenting Inside the CERT Oracle Secure Coding Standard for Java [CON2368] at JavaOne 2014. David Svoboda and Yozo Toda, lead analyst at the JPCERT Coordination Center, will be presenting Anatomy of Another Java Zero-Day Exploit [CON2120].

We continue to perform Source Code Analysis Laboratory (SCALe) assessments, which has led to a smattering of improvements to The CERT Oracle Secure Coding Standard for Java as we evolve rules to be clearer and more precise and to simplify conformance.

How are you using the CERT Secure Coding Standards?

As a reader of this eNewsletter, we want to hear from you. Submit your comments about how you are using CERT Secure Coding Standards.

Language Standards Updates

CERT C Coding Standard

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

Added

- EXP46-C. Do not use a bitwise operator with a Boolean-like operand Was previously EXP17-C. Do not use a bitwise operator in place of logical operator, or vice versa; updated normative text.

Changed

- INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data
  Clarified that this rule also applies to library functions that perform the conversions "out-of-sight."
- INT34-C. Do not shift an expression by a negative number of bits or by greater than or equal to the number of bits that exist in the operand
  Updated the implementation details section to be right-shift-specific.

Removed

- EXP17-C. Do not use a bitwise operator in place of logical operator, or vice versa

CERT C++ Secure Coding Standard

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

Added

- EXP40-CPP. Do not pass a reference or nontrivially-copyable type to va_start
- MSC36-CPP. Value-returning functions must return a value from all exit paths
- MSC37-CPP. Do not return from a function declared [[noreturn]]
- DCL39-CPP. Functions declared with [[noreturn]] must return void
- Relation to the CERT C Coding Standard
  Added a child page to describe how the CERT C++ Secure Coding Standard relates to the CERT C Coding Standard.

Changed

- The "Arrays and the STL" section has been renamed to "Containers"-, and its designation has changed from ARR to CTR. All guideline titles in this section have been updated to reflect these changes.
- DCL33-CPP. Never qualify a reference type with const or volatile
  Considerable edits to content, including title change.
- DCL37-CPP. Overloaded postfix increment and decrement operators should return a const object
  Considerable edits to content, including title change.
- ERR37-CPP. Honor exception specifications
  Considerable edits to content to update it for noexcept specifications; title change.
- STR08-CPP. Do not specify the bound of a character array initialized with a string literal
  Was previously STR36-CPP. Do not specify the bound of a character array initialized with a string literal; this guideline was downgraded from a rule to a recommendation and matches the severity of the analogous C guideline.
- FIO19-CPP. Do not create temporary files in shared directories
  Was previously FIO43-CPP. Do not create temporary files in shared directories; this guideline was downgraded from a rule to a recommendation and matches the severity of the analogous C guideline.
- INT18-CPP. Evaluate integer expressions in a larger size before comparing or assigning to that size
  Was previously INT35-CPP. Evaluate integer expressions in a larger size before comparing or assigning to that size; this guideline was downgraded from a rule to a recommendation and matches the severity of the analogous C guideline.
- FLP05-CPP. Convert integers to floating point for floating point operations
  Was previously FLP33-CPP. Convert integers to floating point for floating point operations; this guideline was downgraded from a rule to a recommendation and matches the severity of the analogous C guideline.
- DCL20-CPP. Use volatile for data that cannot be cached

Was previously DCL34-CPP. Use volatile for data that cannot be cached; this guideline was downgraded from a rule to a recommendation and matches the severity of the analogous C guideline.

- INT30-CPP. Do not cast to an out-of-range enumeration value
  Considerable edits to content, including title change.
- CTR32-CPP. Do not use iterators invalidated by container modification and MEM30-CPP. Do not access freed memory
  Received NCCE/CS pairs from STR33-CC. Do not access invalid output of c_str() or data().
- MEM32-CPP. Detect and handle memory allocation errors
  Considerable edits to the content; added and extended examples from MEM36-CPP.
- MEM34-CPP. Only free memory allocated dynamically
  Considerable edits to the content. Received NCCE/CS pair and text from MEM31-CPP. Free dynamically allocated memory exactly once.
- MEM30-CPP. Do not access freed memory
  Considerable edits to the content.
- OOP38-CPP. Gracefully handle self-assignment
  Considerable edits to the content, including a title change. Was previously MEM42-CPP. Ensure that copy assignment operators do not damage an object that is copied to itself.
- ERR40-CPP. Do not leak resources when handling exceptions
  Considerable edits to content. It received the content of MEM33-CPP. Ensure that aborted constructors do not leak, and was moved from the MEM section into the ERR section. Was previously MEM44-CPP. Do not leak resources when handling exceptions.

Removed

- DCL39-CPP. Non-const references should only be initialized with lvalues
  This rule affects only C++03 and earlier and is ill-formed code in C++11 and later.
- FLP31-CPP. Do not call functions expecting real values with complex values
  This rule applies only to ill-formed C++ code.
- STR33-CPP. Do not access invalid output of c_str() or data()
  This rule affects only C++03 or is covered by CTR32-CPP. Do not use iterators invalidated by container modification and MEM30-CPP. Do not access freed memory.
- MEM36-CPP. Never allocate more than one resource in a single statement
  This rule was subsumed by MEM32-CPP. Detect and handle memory allocation errors
- MEM33-CPP. Ensure that aborted constructors do not leak
  This rule was subsumed by ERR40-CPP. Do not leak resources when handling exceptions
- MEM31-CPP. Free dynamically allocated memory exactly once
  This rule was subsumed by MEM34-CPP. Only free memory allocated dynamically

The following rules were removed as being covered by the CERT C Coding Standard:

- PRE31-CPP. Avoid side-effects in arguments to unsafe macros
- EXP31-CPP. Avoid side-effects in assertions
- EXP34-CPP. Ensure a null pointer is not dereferenced

- EXP37-CPP. Call variadic functions with the arguments intended by the API
- EXP38-CPP. Do not modify constant values
- ARR31-CPP. Use consistent array notation across all source files
- STR30-CPP. Do not attempt to modify string literals
- STR31-CPP. Guarantee that storage for character arrays has sufficient space for character data and the null terminator
- STR32-CPP. Null-terminate character arrays as required
- STR34-CPP. Cast characters to unsigned types before converting to larger integer sizes
- STR37-CPP. Arguments to character handling functions must be representable as an unsigned char
- MEM35-CPP. Allocate sufficient memory for an object
- FIO32-CPP. Do not perform operations on devices that are only appropriate for files
- FIO33-CPP. Detect and handle input output errors resulting in undefined behavior
- FIO38-CPP. Do not use a copy of a FILE object for input and output
- FIO34-CPP. Use int to capture the return value of character IO functions
- FIO35-CPP. Use feof() and ferror() to detect end-of-file and file errors when sizeof(int) == sizeof(char)
- FIO36-CPP. Do not assume a new-line character is read when using fgets()
- FIO37-CPP. Do not assume character data has been read
- FIO40-CPP. Reset strings on fgets() failure
- FIO41-CPP. Do not call getc() or putc() with stream arguments that have side effects
- FIO44-CPP. Only use values for fsetpos() that are returned from fgetpos()
- ENV30-CPP. Do not modify the string returned by getenv()
- ENV31-CPP. Do not rely on an environment pointer following an operation that may invalidate it
- ENV32-CPP. All atexit handlers must return normally
- ERR32-CPP. Do not rely on indeterminate values of errno
- MSC31-CPP. Ensure that return values are compared against the proper type
- INT30-CPP. Ensure that unsigned integer operations do not wrap
- INT31-CPP. Ensure that integer conversions do not result in lost or misinterpreted data
- INT32-CPP. Ensure that operations on signed integers do not result in overflow
- INT33-CPP. Ensure that division and modulo operations do not result in divide-by-zero errors
- INT34-CPP. Do not shift a negative number of bits or more bits than exist in the operand
- FLP30-CPP. Do not use floating point variables as loop counters
- FLP32-CPP. Prevent or detect domain and range errors in math functions
- FLP34-CPP. Ensure that floating point conversions are within range of the new type
- FLP36-CPP. Beware of precision loss when converting integral types to floating point

CERT Oracle Secure Coding Standard for Java

Editors: Adam O'Brien (Oracle), David Svoboda (SEI/CERT)

Added

The following new rules are all currently stubs. We will flesh them out soon.

- JNI02-J. Do not assume object references are constant or unique
- JNI03-J. Do not use direct pointers to Java objects in JNI code
- JNI04-J. Do not assume that Java strings are null-terminated

- EXP07-J. Prevent loss of useful data due to weak references
- FIO15-J. Do not reset a servlet's output stream after committing it
- IDS14-J. Do not trust the contents of hidden form fields
- IDS15-J. Do not allow sensitive information to leak outside a trust boundary
- MSC08-J. Do not store non-serializable objects as attributes in an HTTP session
- NUM14-J. Use shift operators correctly
- OBJ12-J. Respect object-based annotations
- VNA06-J. Do not use non-static member fields in a servlet

Changed

- JNI01-J. Safely invoke standard APIs that perform tasks using the immediate caller's class loader instance (loadLibrary) has new code examples illustrating compliance and noncompliance.
- LCK08-J. Ensure actively held locks are released on exceptional conditions has a new noncompliant code example that unlocks a lock that might not have been locked. That would be unlocky (smile)
- SER07-J. Do not use the default serialized form for classes with implementation-defined invariants now has code examples that illustrate CVE-2012-0507, which exploited the `AtomicReferenceArray` class in Java 1.7.0_02. OBJ06-J. Defensively copy mutable inputs and mutable internal components also mentions this vulnerability, as it was mitigated by complying with this rule, too.
- OBJ03-J. Prevent heap pollution now has code examples to illustrate lists of lists, including variadic parameters.
- EXP02-J. Do not use the Object.equals() method to compare two arrays has a new title, updated descriptions, and new noncompliant code examples and compliant solutions.

Deprecation Candidates

- NUM04-J. Do not use floating-point numbers if precise computation is required is conditional on programmer's intent.
- NUM05-J. Do not use denormalized numbers is unenforcable.
- NUM06-J. Use the strictfp modifier for floating-point calculation consistency across platforms is conditional on programmer's intent.

Removed

FIO15-J. Do not operate on untrusted file links

CERT Secure Coding Standard for Android

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

Added

- DRD20-J. Specify permissions when creating files via the NDK
- Analysis of Android Applicability: The CERT C Coding Standard (Rules and Recommendations) is a new table showing the CERT C Coding Standard rules and recommendations and stating their applicability to the development of Android applications, according to an initial analysis.
- Analysis of Android Applicability: the CERT Oracle Secure Coding Standard for Java (Rules): Analyses were added with respect to new Java rules JNI02-J, JNI03-J, and JNI04-J n

Changed

- DRD03-J. Do not broadcast sensitive information using an implicit intent has new reference to "intent sniffing."
- DRD13-J. Do not provide addJavascriptInterface method access in a WebView which could contain untrusted content. (API level JELLY_BEAN or below) was scoped and titled to clarify that it is about Web Views in mobile apps.
- Analysis of Android Applicability: CERT's Java Coding Guidelines: Analyses with respect to Java guidelines 16, 17, and 19 were modified with text augmenting the guideline as written in the book so that the applicability to Android of the guideline is explained.

CERT Perl Secure Coding Standard

Editor: David Svoboda (SEI/CERT)

No Perl rules were added, removed, deprecated, or substantively changed in July and August.

Upcoming Events and Training

Conference:
Protect 2014 - September 8-11, Washington Hilton, Washington, D.C.
Robert Seacord is presenting TB3335 - Why are we still not programming securely?

Conference:
Java One - September 28-October 2, 2014 - San Francisco, CA
David Svoboda and Robert Seacord are presenting Inside the CERT Oracle Secure Coding Standard for Java [CON2368].
David Svoboda and Yozo Toda are presenting Anatomy of Another Java Zero-Day Exploit [CON2120].

Conference:
TSP Symposium - November 3-6, 2014
The Team Software Process (TSP) Symposium 2014 technical program will go beyond the core methodology of TSP to encompass a broader range of complementary practices that contribute to peak performance on system and software projects.

The unifying theme of the conference is quality. Ultimately, a quality product and service must be delivered on time and within budget, be secure, be sustainable, and provide value to end users.

Conference:
48th Annual Hawaii International Conference on System Sciences - January 5-8, 2015 Grand Hyatt Kauai, Hawaii

Recently Released:
These programs are for individuals in government and industry organizations that are looking to build, assess, or evaluate an insider threat program while protecting the privacy and civil liberties of their employees.

Our People

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

David Svoboda has been the primary developer on a diverse set of software development projects at Carnegie Mellon University since 1991. His projects have ranged from hierarchical chip modeling and social organization simulation to automated machine translation (AMT). His KANTOO AMT software, developed in 1996, is still in production use at Caterpillar. He has over 13 years of Java development experience, starting with Java 2, and his Java projects include Tomcat servlets and Eclipse plug-ins. He has taught Secure Coding in C and C++ all over the world to various groups in the military, government, and banking industries.

Secure Coding Resources

Listen to *Raising the Bar - Mainstreaming CERT C Secure Coding Rules*

by Robert C. Seacord and Julia H. Allen

Read *Performance of Compiler-Assisted Memory Safety Checking*
by David Keaton

Read *Two Secure Coding Tools for Analyzing Android Apps*
by Will Klieber and Lori Flynn

Subscribe to Our eNewsletter

Join the SEI CERT Secure Coding Community