December 2013/January 2014 Winter Edition

News

I hope everyone enjoyed their holidays. My present this year was that we completed the manuscript for *The CERT C Coding Standard* (pictured at left).

The final manuscript weighed in at 98 rules and not the predicted 92, as the new cover shows. Other than that, we are quite happy with the quality of the manuscript and are looking forward to the release of the book in April, 2014. Work is continuing on the recommendations on the wiki, which are not included in the book. If you would like to contribute to this or other efforts, and want to contact us privately, please send email to secure-coding@cert.org.

In other news, ISO/IEC *TS 17961:2013(E), Information Technology - Programming Languages, Their Environments and System Software Interfaces - C Secure Coding Rules* [ISO/IEC TS 17961:2013] was officially published in November 2013 and is available for purchase at the ISO store (http://www.iso.org/iso/catalogue_detail.htm?csnumber=61134). The purpose of ISO/IEC TS 17961 is to establish a baseline set of requirements for analyzers, including static analysis tools and C language compilers, to be applied by vendors that wish to diagnose insecure code beyond the requirements of the language standard. All rules are meant to be enforceable by static analysis. The criterion for selecting these rules is that analyzers that implement these rules must be able to effectively discover secure coding errors without generating excessive false positives.

We asked, You answered.

How are you using the CERT Secure Coding Standards?

"I have taken the Secure Coding for C/C++ - Integers course. Thank you for preparing this wonderful course! It is the best teaching material I have read in recent years. I haven't taken the Strings course yet, but I am sure it's at the same quality. The language of the course and examples are clear. Course doesn't leave room for ambiguity or confusion. There are references to the standard. The language invites engineers to read, because it is precise, direct and reassures that the author knows what he is talking about." - Ismail

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 Secure Coding Standard

Editors: Martin Sebor (Cisco Systems), Aaron Ballman (SEI)

Guidelines Added

MSC40-C. Do not violate constraints
INT35-C. Use correct integer precisions
ENV33-C. Do not call system()
FIO45-C. Avoid TOCTOU race conditions while accessing files
FIO46-C. Do not access a closed file
FIO23-C. Do not exit with unflushed data in stdout or stderr

Guidelines Changed

ARR39-C. Do not add or subtract a scaled integer to a pointer was moved during the course of review; it was formerly EXP41-C.
ARR36-C. Do not subtract or compare two pointers that do not refer to the same array has improved code examples. The noncompliant code example takes the address of the local

variable and subtracts the array address from it (which will produce the expected behavior on some platforms, but is still forbidden). The compliant code example subtracts from the pointer past the end of the array. Also, the text was changed to refer to element counts rather than byte counts (which contributed nothing to the rule).
ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
New noncompliant and compliant code examples were added to illustrate addition to a null pointer, as was done by the Mark Dowd flash vulnerability. See http://www.securityfocus.com/blogs/746 for more information. The code examples that involved a `skip` variable were transferred to ARR38-C. Guarantee that library functions do not form invalid pointers and the code examples involving the `fread()` system call were transferred from ARR30-C to ARR39-C. Do not add or subtract a scaled integer to a pointer.
CON09-C. Avoid the ABA problem when using lock-free algorithms was moved during the course of review; it was formerly CON39-C.
ENV00-C was moved from a recommendation to a rule; it is now ENV34-C. Do not store pointers returned by certain functions.
ENV04-C was moved from a recommendation to a rule; it is now ENV33-C. Do not call system() if you do not need a command processor.
ERR30-C. Set errno to zero before calling a library function known to set errno, and check errno only after the function returns a value indicating failure has lost its `signal()` and `setlocale()` code examples, as they clearly violate ERR33-C. Detect and handle standard library errors.
EXP44-C. Do not use side effects in operands to sizeof, _Alignof, or _Generic This rule was moved during the course of review; it was formerly EXP06-C.
EXP18-C has been moved from a recommendation to a rule; it is now EXP45-C. Do not perform assignments in selection statements
Dynamic allocation content was moved from MEM09-C. Do not assume memory allocation functions initialize memory to EXP33-C. Do not read uninitialized memory.
EXP37-C. Call functions with the correct number and type of arguments The 1st set of code examples was improved to more precisely illustrate that the problem is not with parameter-less function prototype. Due to these improvements, the second NCCE/CS is completely redundant, so it was eliminated. Finally, the third set of code samples, which dealt with a variadic function, actually violated DCL40-C. Do not create incompatible declarations of the same function or object, so it was moved there.
In EXP39-C. Do not access a variable through a pointer of an incompatible type, we deleted code examples that tried to access a float that was unioned with an `int` that got modified. This works on some machines because of type punning, but it is not guaranteed by C11.
FLP36-C. Preserve precision when converting integral values to floating-point type now uses `PRECISION()` and cites INT35-C. Use correct integer precisions.
FIO21-C. Do not create temporary files in shared directories was moved during the course of review; it was formerly FIO43-C.
FIO34-C. Distinguish between characters read from a file and EOF or WEOF has assimilated the old FIO34-C and FIO35-C and included examples dealing with wide characters.
INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data has a new exception to permit conversion of characters between different character types.
INT32-C. Ensure that operations on signed integers do not result in overflow has several changes:

> Its division and modulo code examples are now distinct from those in INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors.
> Its left-shift examples are now distinct from those in INT34-C. Do not shift a negative number of bits or more bits than exist in the operand.
> Finally, it has a mere paragraph describing the behavior of atomic integers- rather than

code examples.
INT30-C. Ensure that unsigned integer operations do not wrap now has a single paragraph describing the behavior of atomic integers- rather than code examples. Also, exception 3 allows wrapping on the left-shift operator- and references INT34-C. Do not shift a negative number of bits or more bits than exist in the operand.
INT11-C was changed to a rule; it is now INT36-C. Converting a pointer to integer or integer to pointer.
MEM31-C. Free dynamically allocated memory when no longer needed has a new exception (based on the formerly final compliant solution). Memory need not be freed if it can be referenced from static variables.
MEM35-C. Allocate sufficient memory for an object has swallowed EXP01-C. Do not take the size of a pointer to determine the size of the pointed-to type. It has new "normative" text saying to be sure to use `sizeof` on the right type.
DCL41-C. Do not declare variables inside a switch statement before the first case label has a new title- but no other changes.
In MSC38-C. Do not treat a predefined identifier as an object if it might only be implemented as a macro we updated the wording and replaced one NCCE/CS pair involving `assert()` with a different example.
PRE31-C. Do not perform side effects in arguments to unsafe macros has a compliant solution demonstrating the benefits of using a `_Generic` selection expression.
SIG31-C. Do not access shared objects in signal handlers we updated the wording and added NCCEs, a CS, and an exception.
STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator now has code examples dealing with `fscanf()`.
STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string is more limited; it is violated only if the non-null-terminated character sequence is passed to a library function that expects a null-terminated byte string.
In FLP32-C. Prevent or detect domain and range errors in math functions we updated the `math_errhandling` examples to not presume usage of a macro.

Guidelines Deprecated & Removed
FLP31-C. Do not call functions expecting real values with complex values was merged into EXP37-C. Call functions with the correct number and type of arguments.
MEM09-C. Do not assume memory allocation functions initialize memory was subsumed by EXP33-C. Do not read uninitialized memory.
EXP31-C. Do not perform side effects in assertions was subsumed by PRE31-C. Do not perform side effects in arguments to unsafe macros.
ARR34-C. Ensure that array types in expressions are compatible was subsumed by EXP39-C. Do not access a variable through a pointer of an incompatible type.
STR35-C. Do not copy data from an unbounded source to a fixed-length array was merged into STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator.
STR36-C was converted from a rule to a recommendation: STR11-C. Do not specify the bound of a character array initialized with a string literal. The severity is now Low; typically violations of this recommendation result in only a limited buffer overflow that no attacker can control or modify.

CERT C++ Secure Coding Standard
Editors: Martin Sebor (Cisco Systems), Aaron Ballman (SEI)

No C++ rules were added, removed, deprecated, or substantively changed last month.

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

Guidelines Changed

No Java rules were were added, removed, deprecated, or substantively changed last month.

CERT Perl Secure Coding Standard
Editor: David Svoboda (SEI)

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

Upcoming Events and Training

FloCon
January 13-16, 2014
FloCon, a network security conference, takes place at the Francis Marion Hotel in Charleston, South Carolina. This open conference provides a forum for operational network analysts, tool developers, researchers, and other parties interested in the analysis of large volumes of traffic to showcase the next generation of flow-based analysis techniques. http://www.cert.org/flocon/

CERT� Operational Resilience: MANAGE, PROTECT, AND SUSTAIN [Virtual Event]
January 23, 2014, http://www.sei.cmu.edu/goto/cert-operational-resilience

Our People

Each month, we highlight staff members behind our secure coding research. This month we feature Lori Flynn.

Lori Flynn is a Software Security Engineer at CERT. Her ongoing work includes an Android-focused project involving development of new secure coding rules and composable static analysis of apps to check for compliance with data flow rules. Lori's past experience includes network security research, system analysis, research prototype development, standards-based security analyses including ISO 27001 and ITU X.805, and collaboration on a novel static analysis method for polymorphic program detection that resulted in a patent. Her PhD research focused on secure multicast routing protocols for ad hoc mobile networks.

Secure Coding Resources

Read *Secure Design Patterns* by Chad Dougherty, Kirk Sayre, Robert C. Seacord, David Svoboda, and Kazuya Togashi. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=9115

Listen to *Raising the Bar: Mainstreaming CERT C Secure Coding Rules* by Robert C. Seacord http://www.cert.org/podcast/mp3/2/20140107seacord-full.mp3

Watch Robert Seacord discuss *Secure Coding - Avoiding Future Security Incidents,* http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=54982

Subscribe to Our eNewsletter

Join the SEI CERT Secure Coding Community