March 15, 2012

Changes in WG14 Document from v. N1579 to v. N1609

Rule Identifiers

The following rule identifiers were adopted to replace original *C Secure Coding* identifiers such as (MEM30-C) and (SIG31-C):

Identifier	Rule
`ptrcomp`	5.1 Accessing an object through a pointer to an incompatible type
`accfree`	5.2 Accessing freed memory
`accsig`	5.3 Accessing shared objects in signal handlers
`cntradd`	5.4 Adding or subtracting a byte count integer to an element pointer
`cndasgn`	5.5 Assigning in conditional expressions
`asyncsig`	5.6 Calling functions in the C standard library other than abort, _Exit, and signal from within a signal handler
`argcomp`	5.7 Calling functions with incorrect arguments
`sigcall`	5.8 Calling signal from interruptible signal handlers
`syscall`	5.9 Calling system
`funcaddr`	5.10 Comparing function addresses to zero
`padcomp`	5.11 Comparison of padding data
`intptrconv`	5.12 Converting a pointer to integer or integer to pointer
`alignconv`	5.13 Converting pointer values to more strictly aligned pointer types
`filecpy`	5.14 Copying a `FILE` object
`funcdecl`	5.15 Declaring the same function or object in incompatible ways
`nullref`	5.16 Dereferencing a null pointer
`divzero`	5.17 Dividing by zero
`addrescape`	5.18 Escaping of the address of an automatic object
`signconv`	5.19 Conversion of signed characters to wider integer types
`swtchdflt`	5.20 Use of an implied default in a switch statement
`fileclose`	5.21 Failing to close files or free dynamic memory when they are no longer needed
`liberr`	5.22 Failing to detect and handle standard library errors
`libptr`	5.23 Forming invalid pointers by library function
`invptr`	5.24 Forming or using out-of-bounds pointers or array subscripts
`dblfree`	5.25 Freeing memory multiple times
`usrfmt`	5.26 Including tainted or out-of-domain input in a format string
`inverrno`	5.27 Incorrectly setting and using errno
`ioileave`	5.28 Interleaving stream inputs and outputs without a flush or positioning call
`strmod`	5.29 Modifying string literals
`libmod`	5.30 Modifying the string returned by getenv, localeconv, setlocale, and strerror
`intoflow`	5.31 Overflowing signed integers
`chrsgnext`	5.32 Passing arguments to character handling functions that are not representable as unsigned
`restrict`	5.33 Passing pointers into the same object as arguments to different restrict-qualified parameters
`xfree`	5.34 Reallocating or freeing memory that was not dynamically allocated
`uninitref`	5.35 Referencing uninitialized memory
`ptrobj`	5.36 Subtracting or comparing two pointers that do not refer to the same array
`taintstrcpy`	5.37 Tainted strings are passed to a string copying function

`sizeofptr`	5.38 Taking the size of a pointer to determine the size of the pointed-to type
`taintnoproto`	5.39 Using a tainted value as an argument to an unprototyped function pointer
`taintformatio`	5.40 Using a tainted value to write to an object using a formatted input or output function
`xfilepos`	5.41 Using a value for fsetpos that is returned from fgetpos
`libuse`	5.42 Using an object overwritten by getenv, localeconv, setlocale, and strerror
`chreof`	5.43 Using character values that are indistinguishable from EOF
`resident`	5.44 Using identifiers that are reserved for the implementation
`invfmtstr`	5.45 Using invalid format strings
`taintsink`	5.46 Tainted, potentially mutilated, or out-of-domain integer values are used in a taintedness sink

Changes to Text

General

- All cross references to C99 were updated to C11
- Wording such as "C99-conforming" and "C11-conforming" was changed to C-conforming
- All UB references were updated from C99 to C11
- References to "tainted sink" were changed to "restricted sink"

Introduction

Significantly modified; new text added

Section 1, Scope

Final paragraph deleted

Section 2, Conformance

Significantly modified; new text added

Section 4.0, Terms and Definitions

- Definitions were reworded to conform with ISO style.
- The following table shows new terms in *italics* and deleted terms in ~~strikethrough~~.

N1609 (current)	**N1570 (original)**
4.1 analyzer	4.1 analyzer
4.2 asynchronous-safe function; *asynchronous-signal safe*	4.2 asynchronous-safe
4.3 data flow *analysis*	4.3 data flow
4.4 dereferenceable pointer	4.4 dereferenceable pointer
4.5 derived type	4.5 derived type
4.6 exploit	4.6 exploit
4.7 mutilated	~~4.7 invalid pointer~~
4.8 out-of-domain value	~~4.8 non-dereferenceable pointer~~

4.9 persistent signal handler	4.9 out-of-domain value
4.10 *restricted sink*	4.10 persistent signal handler
4.11 sanitize	4.11 sanitize
4.12 security flaw	4.12 security flaw
4.13 security policy	4.13 security policy
4.14 static analysis	4.14 static analysis
4.15 tainted *value*	4.15 tainted ~~data~~
4.16 *target implementation*	~~4.16 taintedness sink~~
4.17 *UB*	4.17 untrusted data
4.18 *unexpected behavior*	4.18 valid pointer
4.19 *unsigned integer wrapping*	4.19 vulnerability
4.20 untrusted data	
4.21 valid pointer	
4.22 vulnerability	

Section 5.0, Rules

The following table lists rules whose text and/or code examples have been modified.

Rule	**Modification**
5.1 Accessing an object through a pointer to an incompatible type [ptrcomp]	slightly modified Ex. 1 code; deleted Ex. 2
5.2 Accessing freed memory [accfree]	slightly modified Ex. 2 code
5.3 Accessing shared objects in signal handlers [accsig]	modified code example
5.4 Adding or subtracting a byte count to an element pointer [cntradd]	modified text; modified code, Ex. 4
5.5 No assignment in conditional expressions [boolasgn]	inserted `...while` in third bullet (to read `do...while`)
5.6 Calling functions in the C standard library other than abort, _Exit, and signal from within a signal handler [asyncsig]	slightly modified Ex. 1, 2, 3 code
5.7 Calling functions with incorrect arguments [argcomp]	slightly modifed Ex. 2 code
5.8 Calling signal from interruptible signal handlers [sigcall]	modified para 1
5.9 Calling system [syscall]	
5.10 Comparing function addresses to zero [funcaddr]	modified Ex. 1 (changed `getuid` and `geteuid` to `thrd_current`)
5.11 Comparison of padding data [padcomp]	
5.12 Converting a pointer to integer or integer to pointer [intptrconv]	
5.13 Converting pointer values to more strictly aligned	slightly modified Ex. 1 & 2 code

pointer types [alignconv]	
5.14 Copying a FILE object [filecpy]	slightly modified code example
5.15 Declaring the same function or object in incompatible ways [funcdecl]	deleted note 2
5.16 Dereferencing an out-of-domain pointer [nullref]	slightly modified text and code example
5.17 Dividing by zero [divzero]	slightly modified Ex. 1 & 2 text and code
5.18 Escaping of the address of an automatic object [addrescape]	modified Ex. 1 & 3code
5.19 Conversion of signed characters to wider integer types [signconv]	modified code example
5.20 Use of an implied default in a switch statement [swtchdflt]	modified text
5.21 Failing to close files or free dynamic memory when they are no longer needed [fileclose]	significantly modified text, added new text; modified Ex. 1 & 2 code
5.22 Failing to detect and handle standard library errors [liberr]	modified code example
5.23 Forming invalid pointers by library function [libptr]	significantly modified text, added new text; significantly modified example text
5.24 Forming or using out-of-bounds pointers or array subscripts [invptr]	modified Ex 1–5 and 11 code; deleted Ex. 12–15
5.25 Freeing memory multiple times [dblfree]	slightly modified Ex. 2 code
5.26 Including tainted or out-of-domain input in a format string [usrfmt]	slightly modified Ex. 3 code
5.27 Incorrectly setting and using errno [inverrno]	5.27.1 Table 5 is now Table 4 5.27.2 Table 6 is now Table 5 5.27.3 slightly modified para. 1 5.27.4 slightly modified Ex. 1 code
5.28 Interleaving stream inputs and outputs without a flush or positioning call [ioileave]	
5.29 Modifying string literals [strmod]	slightly modified Ex. 1 code; slightly modified Ex. 2 text and code; slightly modified code in Exception section
5.30 Modifying the string returned by getenv, localeconv, setlocale, and strerror [libmod]	slightly modified Ex. 1 code
5.31 Overflowing signed integers [intoflow]	significantly modified text; deleted para 1; deleted Table 7; modified Ex. 1 & 2 code
5.32 Passing arguments to character-handling functions that are not representable as unsigned char [chrsgnext]	
5.33 Passing pointers into the same object as arguments to different restrict-qualified parameters [restrict]	slightly modified Ex. 1 code
5.34 Reallocating or freeing memory that was not dynamically allocated [xfree]	slightly modified Ex. 1 & 2 code

5.35 Referencing uninitialized memory [uninitref]	modified Ex. 2, 3, 4 code
5.36 Subtracting or comparing two pointers that do not refer to the same array [ptrobj]	slightly modified text and code (changed `string` to `c_str`)
5.37 Tainted strings are passed to a string copying function [taintstrcpy]	New
5.38 Taking the size of a pointer to determine the size of the pointed-to type [sizeofptr]	significantly modified text; significantly modified code example
5.39 Using a tainted value as an argument to an unprototyped function pointer [taintnoproto]	New
5.40 Using a tainted value to write to an object using a formatted input or output function [taintformatio]	New
5.41 Using a value for fsetpos other than a value returned from fgetpos [xfilepos]	
5.42 Using an object overwritten by getenv, localeconv, setlocale, and strerror [libuse]	
5.43 Using character values that are indistinguishable from EOF [chreof]	
5.44 Using identifiers that are reserved for the implementation [resident]	slightly modified Ex. 4–9 code
5.45 Using invalid format strings [invfmtstr]	
5.46 Tainted, potentially mutilated, or out-of-domain integer values are used in a restricted sink [taintsink]	modified para 1; modified Ex. 1 & 2 code

Deleted Rules

The following rules that appeared in N1579 were deleted from N1609.

5.4 Accessing volatile objects through a non-volatile pointer (EXP32-C)

5.7 Assigning in controlling expressions (EXP15-C)

5.8 Assuming a positive remainder when using the % operator (INT10-C)

5.9 Assuming character data does not contain a null byte (FIO37-C)

5.10 atexit-registered handler does not return (ENV32-C)

5.16 Comparing or assigning expressions to a larger size objects (INT35-C)

5.19 Converting floating point values to types that cannot represent their value (FLP34-C)

5.20 Converting integer to a type that is unable to represent its value (INT31-C)

5.23 Declaring an identifier with conflicting linkage classifications (DCL36-C)

5.32 Failing to prevent or detect domain and range errors in math functions (FLP32-C)

5.33 Failing to sanitize the environment when invoking external programs (ENV03-C)

5.40 Invoking an unsafe macro with arguments containing side effects (PRE31-C)

5.41 Modifying constant values (EXP40-C)

5.44 Not finishing case labels with a break statement (MSC17-C)
5.48 Performing bitwise operations on Boolean operands (EXP16-C)
5.51 Shifting signed types (INT13-C)
5.55 Using abort or assert when atexit handlers are registered (ERR06-C)
5.59 Using integer arithmetic to calculate a value for assignment to a floating-point variable (FLP33-C)
5.61 Using non-unique identifiers (DCL32-C)
5.63 Using the sizeof operator on an expression that contains side effects (EXP06-C)
5.64 Using trigraphs (PRE07-C)
5.65 Wrapping unsigned integers (INT30-C)

Annex B

Updated table to C11 undefined behaviors and deleted classification column.