Coverity

This page was automatically generated and should not be edited.

The information on this page was provided by outside contributors and has not been verified by SEI CERT.

The table below can be re-ordered, by clicking column headers.

Tool Version: 2017.07

Checker	Guideline
ALLOC_FREE_MISMATCH	MEM31-C. Free dynamically allocated memory when no longer needed
ALLOC_FREE_MISMATCH	CON30-C. Clean up thread-specific storage
ALLOC_FREE_MISMATCH (needs improvement)	WIN30-C. Properly pair allocation and deallocation functions
ARRAY_VS_SINGLETON	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ARRAY_VS_SINGLETON	ARR37-C. Do not add or subtract an integer to a pointer to a non-array object
ASSERT_SIDE_EFFECT	MSC11-C. Incorporate diagnostic tests using assertions
ASSERT_SIDE_EFFECTS	PRE31-C. Avoid side effects in arguments to unsafe macros
BAD_ALLOC_ARITHMETIC	ARR38-C. Guarantee that library functions do not form invalid pointers
BAD_ALLOC_STRLEN	ARR38-C. Guarantee that library functions do not form invalid pointers
BAD_ALLOC_STRLEN	MEM35-C. Allocate sufficient memory for an object
BAD_CHECK_OF_WAIT_COND	CON41-C. Wrap functions that can fail spuriously in a loop
BAD_COMPARE	EXP16-C. Do not compare function pointers to constant values
BAD_FREE	MEM34-C. Only free memory allocated dynamically
BAD_SHIFT	INT32-C. Ensure that operations on signed integers do not result in overflow
BAD_SHIFT	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
BAD_SIZEOF	ARR38-C. Guarantee that library functions do not form invalid pointers
BAD_SIZEOF	ARR39-C. Do not add or subtract a scaled integer to a pointer
BUFFER_SIZE	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
BUFFER_SIZE	ARR38-C. Guarantee that library functions do not form invalid pointers
BUFFER_SIZE	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
CHAR_IO	FIO34-C. Distinguish between characters read from a file and EOF or WEOF
CHECKED_RETURN	EXP34-C. Do not dereference null pointers
CHECKED_RETURN	POS54-C. Detect and handle POSIX library errors
CHECKED_RETURN	EXP12-C. Do not ignore values returned by functions
CONSTANT_EXPRESSION_RESULT	EXP46-C. Do not use a bitwise operator with a Boolean-like operand
DEADCODE	MSC07-C. Detect and remove dead code
DEADCODE	MSC12-C. Detect and remove code that has no effect or is never executed
DIVIDE_BY_ZERO	INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors
DONT_CALL	ENV33-C. Do not call system()
DONTCALL	MSC30-C. Do not use the rand() function for generating pseudorandom numbers
DONTCALL	POS33-C. Do not use vfork()
EVALUATION_ORDER	EXP30-C. Do not depend on the order of evaluation for side effects
EVALUATION_ORDER	EXP10-C. Do not depend on the order of evaluation of subexpressions or the order in which side effects take place
EVALUATION_ORDER (partial)	CON40-C. Do not refer to an atomic variable twice in an expression

FORWARD_NULL	EXP34-C. Do not dereference null pointers
INTEGER_OVERFLOW	INT30-C. Ensure that unsigned integer operations do not wrap
LOCK	CON01-C. Acquire and release synchronization primitives in the same module, at the same level of abstraction
MISRA 2012 Rule 13.2	CON40-C. Do not refer to an atomic variable twice in an expression
MISRA C 2004 17.2	ARR36-C. Do not subtract or compare two pointers that do not refer to the same array
MISRA C 2004 17.3	ARR36-C. Do not subtract or compare two pointers that do not refer to the same array
MISRA C 2004 Rule 10.x (needs investigation)	FLP36-C. Preserve precision when converting integral values to floating-point type
MISRA C 2004 Rule 11.4	EXP36-C. Do not cast pointers into more strictly aligned pointer types
MISRA C 2004 Rule 11.5	EXP40-C. Do not modify constant objects
MISRA C 2004 Rule 12.3	EXP44-C. Do not rely on side effects in operands to sizeof, _Alignof, or _Generic
MISRA C 2004 Rule 13.4	FLP30-C. Do not use floating-point variables as loop counters
MISRA C 2004 Rule 15.0	DCL41-C. Do not declare variables inside a switch statement before the first case label
MISRA C 2004 Rule 20.1	DCL37-C. Do not declare or define a reserved identifier
MISRA C 2004 Rule 20.2	DCL37-C. Do not declare or define a reserved identifier
MISRA C 2012 18.2	ARR36-C. Do not subtract or compare two pointers that do not refer to the same array
MISRA C 2012 18.3	ARR36-C. Do not subtract or compare two pointers that do not refer to the same array
MISRA C 2012 Rule 8.1	DCL31-C. Declare identifiers before using them
MISRA C 2012 Rule 8.2	EXP37-C. Call functions with the correct number and type of arguments
MISRA C 2012 Rule 8.4	DCL40-C. Do not create incompatible declarations of the same function or object
MISRA C 2012 Rule 8.14	EXP43-C. Avoid undefined behavior when using restrict-qualified pointers
MISRA C 2012 Rule 10.1	STR34-C. Cast characters to unsigned char before converting to larger integer sizes
MISRA C 2012 Rule 10.2	STR34-C. Cast characters to unsigned char before converting to larger integer sizes
MISRA C 2012 Rule 10.3	STR34-C. Cast characters to unsigned char before converting to larger integer sizes
MISRA C 2012 Rule 10.4	STR34-C. Cast characters to unsigned char before converting to larger integer sizes
MISRA C 2012 Rule 11.1	EXP36-C. Do not cast pointers into more strictly aligned pointer types
MISRA C 2012 Rule 11.2	EXP36-C. Do not cast pointers into more strictly aligned pointer types
MISRA C 2012 Rule 11.5	EXP36-C. Do not cast pointers into more strictly aligned pointer types
MISRA C 2012 Rule 11.7	EXP36-C. Do not cast pointers into more strictly aligned pointer types
MISRA C 2012 Rule 11.8	EXP32-C. Do not access a volatile object through a nonvolatile reference
MISRA C 2012 Rule 14.1	FLP30-C. Do not use floating-point variables as loop counters
MISRA C 2012 Rule 16.1	DCL41-C. Do not declare variables inside a switch statement before the first case label
MISRA C 2012 Rule 17.3	EXP37-C. Call functions with the correct number and type of arguments
MISRA C 2012 Rule 21.1	DCL37-C. Do not declare or define a reserved identifier
MISRA C 2012 Rule 21.2	DCL37-C. Do not declare or define a reserved identifier
MISRA C 2012 Rule 21.5	CON37-C. Do not call signal() in a multithreaded program
MISRA C 2012 Rule 22.5	FIO38-C. Do not copy a FILE object
MISRA C 2012 Rule 22.8	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
MISRA C 2012 Rule 22.8	ERR32-C. Do not rely on indeterminate values of errno
MISRA C 2012 Rule 22.8	ERR33-C. Detect and handle standard library errors
MISRA C 2012 Rule 22.9	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
MISRA C 2012 Rule 22.9	ERR32-C. Do not rely on indeterminate values of errno
MISRA C 2012 Rule 22.9	ERR33-C. Detect and handle standard library errors
MISRA C 2012 Rule 22.10	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
MISRA C 2012 Rule 22.10	ERR32-C. Do not rely on indeterminate values of errno
MISRA C 2012 Rule 22.10	ERR33-C. Detect and handle standard library errors

MISRA_CAST	INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data
MISRA_CAST (needs verification)	FLP34-C. Ensure that floating-point conversions are within range of the new type
MISSING_BREAK	MSC17-C. Finish every set of statements associated with a case label with a break statement
MISSING_LOCK	CON32-C. Prevent data races when accessing bit-fields from multiple threads
MISSING_LOCK (partial)	CON43-C. Do not allow data races in multithreaded code
MISSING_RETURN	MSC37-C. Ensure that control never reaches the end of a non-void function
NEGATIVE_RETURNS	INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data
NEGATIVE_RETURNS	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
NO_EFFECT	MSC12-C. Detect and remove code that has no effect or is never executed
NULL_RETURNS	EXP34-C. Do not dereference null pointers
OPEN_ARGS	FIO03-C. Do not make assumptions about fopen() and file creation
ORDER_REVERSAL	CON35-C. Avoid deadlock by locking in a predefined order
OVERFLOW_BEFORE_WIDEN	INT18-C. Evaluate integer expressions in a larger size before comparing or assigning to that size
OVERRUN	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
OVERRUN	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
PW	EXP40-C. Do not modify constant objects
PW	STR30-C. Do not attempt to modify string literals
PW	STR38-C. Do not confuse narrow and wide character strings and functions
PW	FIO47-C. Use valid format strings
PW.LINKAGE_CONFLICT	DCL36-C. Do not declare an identifier with conflicting linkage classifications
PW. POINTER_CONVERSION_LOSES_BITS	INT36-C. Converting a pointer to integer or integer to pointer
READLINK	POS30-C. Use the readlink() function properly
RESOURCE_LEAK	MEM31-C. Free dynamically allocated memory when no longer needed
RESOURCE_LEAK	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
RESOURCE_LEAK (partial)	FIO42-C. Close files when they are no longer needed
RETURN_LOCAL	DCL30-C. Declare objects with appropriate storage durations
REVERSE_INULL	EXP34-C. Do not dereference null pointers
REVERSE_NEGATIVE	INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data
REVERSE_NEGATIVE	ARR32-C. Ensure size arguments for variable length arrays are in a valid range
SECURE_TEMP	FIO21-C. Do not create temporary files in shared directories
SIZECHECK (deprecated)	MEM35-C. Allocate sufficient memory for an object
STACK_USE	MEM05-C. Avoid large stack allocations
STRING_NULL	STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string
STRING_OVERFLOW	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
STRING_SIZE	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
TAINTED_SCALAR	INT32-C. Ensure that operations on signed integers do not result in overflow
TAINTED_STRING	FIO30-C. Exclude user input from format strings
TAINTED_STRING	STR02-C. Sanitize data passed to complex subsystems
TOCTOU	FIO45-C. Avoid TOCTOU race conditions while accessing files
TOCTOU	POS35-C. Avoid race conditions while checking for the existence of a symbolic link
TOCTOU	FIO01-C. Be careful using functions that use file names for identification
UNINIT	EXP33-C. Do not read uninitialized memory
UNREACHABLE	MSC07-C. Detect and remove dead code
UNREACHABLE	MSC12-C. Detect and remove code that has no effect or is never executed
UNUSED_VALUE	MSC13-C. Detect and remove unused values
USE_AFTER_FREE	MEM30-C. Do not access freed memory

USE_AFTER_FREE	FIO46-C. Do not access a closed file
USE_AFTER_FREE	MEM01-C. Store a new value in pointers immediately after free()
Various concurrency checkers	POS49-C. When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed
VOLATILE_ATOICITY (possible)	CON40-C. Do not refer to an atomic variable twice in an expression