Polyspace Bug Finder

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: R2019a

Checker	Guideline
CERT C: Rec. ARR01-C	ARR01-C. Do not apply the sizeof operator to a pointer when taking the size of an array
CERT C: Rec. ARR02-C	ARR02-C. Explicitly specify array bounds, even if implicitly defined by an initializer
CERT C: Rec. CON01-C	CON01-C. Acquire and release synchronization primitives in the same module, at the same level of abstraction
CERT C: Rec. CON05-C	CON05-C. Do not perform operations that can block while holding a lock
CERT C: Rec. DCL01-C	DCL01-C. Do not reuse variable names in subscopes
CERT C: Rec. DCL02-C	DCL02-C. Use visually distinct identifiers
CERT C: Rec. DCL06-C	DCL06-C. Use meaningful symbolic constants to represent literal values
CERT C: Rec. DCL07-C	DCL07-C. Include the appropriate type information in function declarators
CERT C: Rec. DCL10-C	DCL10-C. Maintain the contract between the writer and caller of variadic functions
CERT C: Rec. DCL11-C	DCL11-C. Understand the type issues associated with variadic functions
CERT C: Rec. DCL12-C	DCL12-C. Implement abstract data types using opaque types
CERT C: Rec. DCL13-C	DCL13-C. Declare function parameters that are pointers to values not changed by the function as const
CERT C: Rec. DCL15-C	DCL15-C. Declare file-scope objects or functions that do not need external linkage as static
CERT C: Rec. DCL16-C	DCL16-C. Use "L," not "l," to indicate a long value
CERT C: Rec. DCL18-C	DCL18-C. Do not begin integer constants with 0 when specifying a decimal value
CERT C: Rec. DCL19-C	DCL19-C. Minimize the scope of variables and functions
CERT C: Rec. DCL22-C	DCL22-C. Use volatile for data that cannot be cached
CERT C: Rec. DCL23-C	DCL23-C. Guarantee that mutually visible identifiers are unique
CERT C: Rec. ENV01-C	ENV01-C. Do not make assumptions about the size of an environment variable
CERT C: Rec. ERR00-C	ERR00-C. Adopt and implement a consistent and comprehensive error-handling policy
CERT C: Rec. EXP00-C	EXP00-C. Use parentheses for precedence of operation
CERT C: Rec. EXP05-C	EXP05-C. Do not cast away a const qualification

CERT C: Rec. EXP08-C	EXP08-C. Ensure pointer arithmetic is used correctly
CERT C: Rec. EXP09-C	EXP09-C. Use sizeof to determine the size of a type or variable
CERT C: Rec. EXP10-C	EXP10-C. Do not depend on the order of evaluation of subexpressions or the order in which side effects take place
CERT C: Rec. EXP11-C	EXP11-C. Do not make assumptions regarding the layout of structures with bit-fields
CERT C: Rec. EXP12-C	EXP12-C. Do not ignore values returned by functions
CERT C: Rec. EXP13-C	EXP13-C. Treat relational and equality operators as if they were nonassociative
CERT C: Rec. EXP19-C	EXP19-C. Use braces for the body of an if, for, or while statement
CERT C: Rec. FIO02-C	FIO02-C. Canonicalize path names originating from tainted sources
CERT C: Rec. FIO11-C	FIO11-C. Take care when specifying the mode parameter of fopen()
CERT C: Rec. FIO21-C	FIO21-C. Do not create temporary files in shared directories
CERT C: Rec. FIO24-C	FIO24-C. Do not open a file that is already open
CERT C: Rec. FLP00-C	FLP00-C. Understand the limitations of floating-point numbers
CERT C: Rec. FLP02-C	FLP02-C. Avoid using floating-point numbers when precise computation is needed
CERT C: Rec. FLP03-C	FLP03-C. Detect and handle floating-point errors
CERT C: Rec. FLP06-C	FLP06-C. Convert integers to floating point for floating-point operations
CERT C: Rec. INT00-C	INT00-C. Understand the data model used by your implementation(s)
CERT C: Rec. INT02-C	INT02-C. Understand integer conversion rules
CERT C: Rec. INT04-C	INT04-C. Enforce limits on integer values originating from tainted sources
CERT C: Rec. INT07-C	INT07-C. Use only explicitly signed or unsigned char type for numeric values
CERT C: Rec. INT08-C	INT08-C. Verify that all integer values are in range
CERT C: Rec. INT09-C	INT09-C. Ensure enumeration constants map to unique values
CERT C: Rec. INT10-C	INT10-C. Do not assume a positive remainder when using the % operator
CERT C: Rec. INT12-C	INT12-C. Do not make assumptions about the type of a plain int bit-field when used in an expression
CERT C: Rec. INT13-C	INT13-C. Use bitwise operators only on unsigned operands
CERT C: Rec. INT14-C	INT14-C. Avoid performing bitwise and arithmetic operations on the same data
CERT C: Rec. INT18-C	INT18-C. Evaluate integer expressions in a larger size before comparing or assigning to that size
CERT C: Rec. MEM00-C	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
CERT C: Rec. MEM01-C	MEM01-C. Store a new value in pointers immediately after free()
CERT C: Rec. MEM02-C	MEM02-C. Immediately cast the result of a memory allocation function call into a pointer to the allocated type
CERT C: Rec. MEM03-C	MEM03-C. Clear sensitive information stored in reusable resources
CERT C: Rec. MEM04-C	MEM04-C. Beware of zero-length allocations
CERT C: Rec. MEM05-C	MEM05-C. Avoid large stack allocations
CERT C: Rec. MEM06-C	MEM06-C. Ensure that sensitive data is not written out to disk
CERT C: Rec. MEM11-C	MEM11-C. Do not assume infinite heap space
CERT C: Rec. MEM12-C	MEM12-C. Consider using a goto chain when leaving a function on error when using and releasing resources
CERT C: Rec. MSC01-C	MSC01-C. Strive for logical completeness

CERT C: Rec. MSC04-C	MSC04-C. Use comments consistently and in a readable fashion
CERT C: Rec. MSC12-C	MSC12-C. Detect and remove code that has no effect or is never executed
CERT C: Rec. MSC13-C	MSC13-C. Detect and remove unused values
CERT C: Rec. MSC15-C	MSC15-C. Do not depend on undefined behavior
CERT C: Rec. MSC17-C	MSC17-C. Finish every set of statements associated with a case label with a break statement
CERT C: Rec. MSC18-C	MSC18-C. Be careful while handling sensitive data, such as passwords, in program code
CERT C: Rec. MSC20-C	MSC20-C. Do not use a switch statement to transfer control into a complex block
CERT C: Rec. MSC21-C	MSC21-C. Use robust loop termination conditions
CERT C: Rec. MSC22-C	MSC22-C. Use the setjmp(), longjmp() facility securely
CERT C: Rec. MSC24-C	MSC24-C. Do not use deprecated or obsolescent functions
CERT C: Rec. POS05-C	POS05-C. Limit access to files by creating a jail
CERT C: Rec. PRE00-C	PRE00-C. Prefer inline or static functions to function-like macros
CERT C: Rec. PRE01-C	PRE01-C. Use parentheses within macros around parameter names
CERT C: Rec. PRE06-C	PRE06-C. Enclose header files in an include guard
CERT C: Rec. PRE07-C	PRE07-C. Avoid using repeated question marks
CERT C: Rec. PRE09-C	PRE09-C. Do not replace secure functions with deprecated or obsolescent functions
CERT C: Rec. STR02-C	STR02-C. Sanitize data passed to complex subsystems
CERT C: Rec. STR03-C	STR03-C. Do not inadvertently truncate a string
CERT C: Rec. STR07-C	STR07-C. Use the bounds-checking interfaces for string manipulation
CERT C: Rec. STR11-C	STR11-C. Do not specify the bound of a character array initialized with a string literal
CERT C: Rec. WIN00-C	WIN00-C. Be specific when dynamically loading libraries
CERT C: Rule ARR30-C	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
CERT C: Rule ARR32-C	ARR32-C. Ensure size arguments for variable length arrays are in a valid range
CERT C: Rule ARR36-C	ARR36-C. Do not subtract or compare two pointers that do not refer to the same array
CERT C: Rule ARR37-C	ARR37-C. Do not add or subtract an integer to a pointer to a non-array object
CERT C: Rule ARR38-C	ARR38-C. Guarantee that library functions do not form invalid pointers
CERT C: Rule ARR39-C	ARR39-C. Do not add or subtract a scaled integer to a pointer
CERT C: Rule CON30-C	CON30-C. Clean up thread-specific storage
CERT C: Rule CON31-C	CON31-C. Do not destroy a mutex while it is locked
CERT C: Rule CON32-C	CON32-C. Prevent data races when accessing bit-fields from multiple threads

CERT C: Rule CON33-C	CON33-C. Avoid race conditions when using library functions
CERT C: Rule CON35-C	CON35-C. Avoid deadlock by locking in a predefined order
CERT C: Rule CON36-C	CON36-C. Wrap functions that can spuriously wake up in a loop
CERT C: Rule CON37-C	CON37-C. Do not call signal() in a multithreaded program
CERT C: Rule CON40-C	CON40-C. Do not refer to an atomic variable twice in an expression
CERT C: Rule CON41-C	CON41-C. Wrap functions that can fail spuriously in a loop
CERT C: Rule CON43-C	CON43-C. Do not allow data races in multithreaded code
CERT C: Rule DCL30-C	DCL30-C. Declare objects with appropriate storage durations
CERT C: Rule DCL31-C	DCL31-C. Declare identifiers before using them
CERT C: Rule DCL36-C	DCL36-C. Do not declare an identifier with conflicting linkage classifications
CERT C: Rule DCL37-C	DCL37-C. Do not declare or define a reserved identifier
CERT C: Rule DCL38-C	DCL38-C. Use the correct syntax when declaring a flexible array member
CERT C: Rule DCL39-C	DCL39-C. Avoid information leakage when passing a structure across a trust boundary
CERT C: Rule DCL40-C	DCL40-C. Do not create incompatible declarations of the same function or object
CERT C: Rule DCL41-C	DCL41-C. Do not declare variables inside a switch statement before the first case label
CERT C: Rule ENV30-C	ENV30-C. Do not modify the object referenced by the return value of certain functions
CERT C: Rule ENV31-C	ENV31-C. Do not rely on an environment pointer following an operation that may invalidate it
CERT C: Rule ENV32-C	ENV32-C. All exit handlers must return normally
CERT C: Rule ENV33-C	ENV33-C. Do not call system()
CERT C: Rule ENV34-C	ENV34-C. Do not store pointers returned by certain functions
CERT C: Rule ERR30-C	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
CERT C: Rule ERR32-C	ERR32-C. Do not rely on indeterminate values of errno
CERT C: Rule ERR33-C	ERR33-C. Detect and handle standard library errors
CERT C: Rule ERR34-C	ERR34-C. Detect errors when converting a string to a number
CERT C: Rule EXP30-C	EXP30-C. Do not depend on the order of evaluation for side effects
CERT C: Rule EXP32-C	EXP32-C. Do not access a volatile object through a nonvolatile reference
CERT C: Rule EXP33-C	EXP33-C. Do not read uninitialized memory
CERT C: Rule EXP34-C	EXP34-C. Do not dereference null pointers
CERT C: Rule EXP36-C	EXP36-C. Do not cast pointers into more strictly aligned pointer types
CERT C: Rule EXP37-C	EXP37-C. Call functions with the correct number and type of arguments

CERT C: Rule EXP39-C	EXP39-C. Do not access a variable through a pointer of an incompatible type
CERT C: Rule EXP40-C	EXP40-C. Do not modify constant objects
CERT C: Rule EXP42-C	EXP42-C. Do not compare padding data
CERT C: Rule EXP43-C	EXP43-C. Avoid undefined behavior when using restrict-qualified pointers
CERT C: Rule EXP44-C	EXP44-C. Do not rely on side effects in operands to sizeof, _Alignof, or _Generic
CERT C: Rule EXP45-C	EXP45-C. Do not perform assignments in selection statements
CERT C: Rule EXP46-C	EXP46-C. Do not use a bitwise operator with a Boolean-like operand
CERT C: Rule EXP47-C	EXP47-C. Do not call va_arg with an argument of the incorrect type
CERT C: Rule FIO30-C	FIO30-C. Exclude user input from format strings
CERT C: Rule FIO32-C	FIO32-C. Do not perform operations on devices that are only appropriate for files
CERT C: Rule FIO34-C	FIO34-C. Distinguish between characters read from a file and EOF or WEOF
CERT C: Rule FIO37-C	FIO37-C. Do not assume that fgets() or fgetws() returns a nonempty string when successful
CERT C: Rule FIO38-C	FIO38-C. Do not copy a FILE object
CERT C: Rule FIO39-C	FIO39-C. Do not alternately input and output from a stream without an intervening flush or positioning call
CERT C: Rule FIO40-C	FIO40-C. Reset strings on fgets() or fgetws() failure
CERT C: Rule FIO41-C	FIO41-C. Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects
CERT C: Rule FIO42-C	FIO42-C. Close files when they are no longer needed
CERT C: Rule FIO44-C	FIO44-C. Only use values for fsetpos() that are returned from fgetpos()
CERT C: Rule FIO45-C	FIO45-C. Avoid TOCTOU race conditions while accessing files
CERT C: Rule FIO46-C	FIO46-C. Do not access a closed file
CERT C: Rule FIO47-C	FIO47-C. Use valid format strings
CERT C: Rule FLP30-C	FLP30-C. Do not use floating-point variables as loop counters
CERT C: Rule FLP34-C	FLP34-C. Ensure that floating-point conversions are within range of the new type
CERT C: Rule FLP37-C	FLP37-C. Do not use object representations to compare floating-point values
CERT C: Rule INT30-C	INT30-C. Ensure that unsigned integer operations do not wrap
CERT C: Rule INT31-C	INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data
CERT C: Rule INT32-C	INT32-C. Ensure that operations on signed integers do not result in overflow
CERT C: Rule INT33-C	INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors
CERT C: Rule INT34-C	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
CERT C: Rule INT35-C	INT35-C. Use correct integer precisions
CERT C: Rule INT36-C	INT36-C. Converting a pointer to integer or integer to pointer
CERT C: Rule MEM30-C	MEM30-C. Do not access freed memory
CERT C: Rule MEM31-C	MEM31-C. Free dynamically allocated memory when no longer needed
CERT C: Rule MEM33-C	MEM33-C. Allocate and copy structures containing a flexible array member dynamically
CERT C: Rule MEM34-C	MEM34-C. Only free memory allocated dynamically
CERT C: Rule MEM35-C	MEM35-C. Allocate sufficient memory for an object
CERT C: Rule MEM36-C	MEM36-C. Do not modify the alignment of objects by calling realloc()

CERT C: Rule MSC07-C	MSC07-C. Detect and remove dead code
CERT C: Rule MSC30-C	MSC30-C. Do not use the rand() function for generating pseudorandom numbers
CERT C: Rule MSC32-C	MSC32-C. Properly seed pseudorandom number generators
CERT C: Rule MSC33-C	MSC33-C. Do not pass invalid data to the asctime() function
CERT C: Rule MSC37-C	MSC37-C. Ensure that control never reaches the end of a non-void function
CERT C: Rule MSC38-C	MSC38-C. Do not treat a predefined identifier as an object if it might only be implemented as a macro
CERT C: Rule MSC39-C	MSC39-C. Do not call va_arg() on a va_list that has an indeterminate value
CERT C: Rule MSC40-C	MSC40-C. Do not violate constraints
CERT C: Rule POS30-C	POS30-C. Use the readlink() function properly
CERT C: Rule POS33-C	POS33-C. Do not use vfork()
CERT C: Rule POS34-C	POS34-C. Do not call putenv() with a pointer to an automatic variable as the argument
CERT C: Rule POS35-C	POS35-C. Avoid race conditions while checking for the existence of a symbolic link
CERT C: Rule POS36-C	POS36-C. Observe correct revocation order while relinquishing privileges
CERT C: Rule POS37-C	POS37-C. Ensure that privilege relinquishment is successful
CERT C: Rule POS38-C	POS38-C. Beware of race conditions when using fork and file descriptors
CERT C: Rule POS39-C	POS39-C. Use the correct byte ordering when transferring data between systems
CERT C: Rule POS44-C	POS44-C. Do not use signals to terminate threads
CERT C: Rule POS48-C	POS48-C. Do not unlock or destroy another POSIX thread's mutex
CERT C: Rule POS49-C	POS49-C. When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed
CERT C: Rule POS51-C	POS51-C. Avoid deadlock with POSIX threads by locking in predefined order
CERT C: Rule POS52-C	POS52-C. Do not perform operations that can block while holding a POSIX lock
CERT C: Rule POS54-C	POS54-C. Detect and handle POSIX library errors
CERT C: Rule PRE30-C	PRE30-C. Do not create a universal character name through concatenation
CERT C: Rule PRE31-C	PRE31-C. Avoid side effects in arguments to unsafe macros
CERT C: Rule PRE32-C	PRE32-C. Do not use preprocessor directives in invocations of function-like macros
CERT C: Rule SIG30-C	SIG30-C. Call only asynchronous-safe functions within signal handlers
CERT C: Rule SIG31-C	SIG31-C. Do not access shared objects in signal handlers
CERT C: Rule SIG34-C	SIG34-C. Do not call signal() from within interruptible signal handlers
CERT C: Rule SIG35-C	SIG35-C. Do not return from a computational exception signal handler
CERT C: Rule STR30-C	STR30-C. Do not attempt to modify string literals
CERT C: Rule STR31-C	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT C: Rule STR32-C	STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string

CERT C: Rule STR34-C	STR34-C. Cast characters to unsigned char before converting to larger integer sizes
CERT C: Rule STR37-C	STR37-C. Arguments to character-handling functions must be representable as an unsigned char
CERT C: Rule STR38-C	STR38-C. Do not confuse narrow and wide character strings and functions
CERT C: Rule WIN30-C	WIN30-C. Properly pair allocation and deallocation functions
CERT-C: Rule EXP35-C	EXP35-C. Do not modify objects with temporary lifetime
CERT-C: Rule FLP32-C	FLP32-C. Prevent or detect domain and range errors in math functions
CERT-C: Rule FLP36-C	FLP36-C. Preserve precision when converting integral values to floating-point type