Parasoft

⚠ 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: 10.4.2

Checker	Guideline
CERT_CPP-CON50-a	CON50-CPP. Do not destroy a mutex while it is locked
CERT_CPP-CON51-a	CON51-CPP. Ensure actively held locks are released on exceptional conditions
CERT_CPP-CON52-a	CON52-CPP. Prevent data races when accessing bit-fields from multiple threads
CERT_CPP-CON53-a	CON53-CPP. Avoid deadlock by locking in a predefined order
CERT_CPP-CON54-a	CON54-CPP. Wrap functions that can spuriously wake up in a loop
CERT_CPP-CON55-a	CON55-CPP. Preserve thread safety and liveness when using condition variables
CERT_CPP-CON56-a	CON56-CPP. Do not speculatively lock a non-recursive mutex that is already owned by the calling thread
CERT_CPP-CTR50-a	CTR50-CPP. Guarantee that container indices and iterators are within the valid range
CERT_CPP-CTR51-a	CTR51-CPP. Use valid references, pointers, and iterators to reference elements of a container
CERT_CPP-CTR52-a	CTR52-CPP. Guarantee that library functions do not overflow
CERT_CPP-CTR53-a	CTR53-CPP. Use valid iterator ranges
CERT_CPP-CTR53-b	CTR53-CPP. Use valid iterator ranges
CERT_CPP-CTR54-a	CTR54-CPP. Do not subtract iterators that do not refer to the same container
CERT_CPP-CTR54-b	CTR54-CPP. Do not subtract iterators that do not refer to the same container
CERT_CPP-CTR55-a	CTR55-CPP. Do not use an additive operator on an iterator if the result would overflow
CERT_CPP-CTR56-a	CTR56-CPP. Do not use pointer arithmetic on polymorphic objects
CERT_CPP-CTR56-b	CTR56-CPP. Do not use pointer arithmetic on polymorphic objects
CERT_CPP-CTR56-c	CTR56-CPP. Do not use pointer arithmetic on polymorphic objects
CERT_CPP-CTR57-a	CTR57-CPP. Provide a valid ordering predicate
CERT_CPP-CTR58-a	CTR58-CPP. Predicate function objects should not be mutable
CERT_CPP-DCL50-a	DCL50-CPP. Do not define a C-style variadic function
CERT_CPP-DCL51-a	DCL51-CPP. Do not declare or define a reserved identifier
CERT_CPP-DCL51-b	DCL51-CPP. Do not declare or define a reserved identifier
CERT_CPP-DCL51-c	DCL51-CPP. Do not declare or define a reserved identifier
CERT_CPP-DCL51-d	DCL51-CPP. Do not declare or define a reserved identifier
CERT_CPP-DCL51-e	DCL51-CPP. Do not declare or define a reserved identifier
CERT_CPP-DCL51-f	DCL51-CPP. Do not declare or define a reserved identifier
CERT_CPP-DCL52-a	DCL52-CPP. Never qualify a reference type with const or volatile
CERT_CPP-DCL53-a	DCL53-CPP. Do not write syntactically ambiguous declarations
CERT_CPP-DCL53-b	DCL53-CPP. Do not write syntactically ambiguous declarations
CERT_CPP-DCL54-a	DCL54-CPP. Overload allocation and deallocation functions as a pair in the same scope

CERT_CPP-DCL55-a	DCL55-CPP. Avoid information leakage when passing a class object across a trust boundary
CERT_CPP-DCL56-a	DCL56-CPP. Avoid cycles during initialization of static objects
CERT_CPP-DCL57-a	DCL57-CPP. Do not let exceptions escape from destructors or deallocation functions
CERT_CPP-DCL57-b	DCL57-CPP. Do not let exceptions escape from destructors or deallocation functions
CERT_CPP-DCL58-a	DCL58-CPP. Do not modify the standard namespaces
CERT_CPP-DCL59-a	DCL59-CPP. Do not define an unnamed namespace in a header file
CERT_CPP-DCL60-a	DCL60-CPP. Obey the one-definition rule
CERT_CPP-ERR50-a	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-b	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-c	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-d	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-e	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-f	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-g	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-h	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-i	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-j	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-k	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-l	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR50-m	ERR50-CPP. Do not abruptly terminate the program
CERT_CPP-ERR51-a	ERR51-CPP. Handle all exceptions
CERT_CPP-ERR51-b	ERR51-CPP. Handle all exceptions
CERT_CPP-ERR52-a	ERR52-CPP. Do not use setjmp() or longjmp()
CERT_CPP-ERR52-b	ERR52-CPP. Do not use setjmp() or longjmp()
CERT_CPP-ERR53-a	ERR53-CPP. Do not reference base classes or class data members in a constructor or destructor function-try-block handler
CERT_CPP-ERR54-a	ERR54-CPP. Catch handlers should order their parameter types from most derived to least derived
CERT_CPP-ERR55-a	ERR55-CPP. Honor exception specifications
CERT_CPP-ERR56-a	ERR56-CPP. Guarantee exception safety
CERT_CPP-ERR57-a	ERR57-CPP. Do not leak resources when handling exceptions
CERT_CPP-ERR58-a	ERR58-CPP. Handle all exceptions thrown before main() begins executing
CERT_CPP-ERR59-a	ERR59-CPP. Do not throw an exception across execution boundaries
CERT_CPP-ERR60-a	ERR60-CPP. Exception objects must be nothrow copy constructible
CERT_CPP-ERR60-b	ERR60-CPP. Exception objects must be nothrow copy constructible
CERT_CPP-ERR61-a	ERR61-CPP. Catch exceptions by lvalue reference
CERT_CPP-ERR61-b	ERR61-CPP. Catch exceptions by lvalue reference
CERT_CPP-ERR62-a	ERR62-CPP. Detect errors when converting a string to a number
CERT_CPP-EXP50-a	EXP50-CPP. Do not depend on the order of evaluation for side effects
CERT_CPP-EXP50-b	EXP50-CPP. Do not depend on the order of evaluation for side effects
CERT_CPP-EXP50-c	EXP50-CPP. Do not depend on the order of evaluation for side effects
CERT_CPP-EXP50-d	EXP50-CPP. Do not depend on the order of evaluation for side effects
CERT_CPP-EXP50-e	EXP50-CPP. Do not depend on the order of evaluation for side effects
CERT_CPP-EXP50-f	EXP50-CPP. Do not depend on the order of evaluation for side effects
CERT_CPP-EXP51-a	EXP51-CPP. Do not delete an array through a pointer of the incorrect type
CERT_CPP-EXP52-a	EXP52-CPP. Do not rely on side effects in unevaluated operands
CERT_CPP-EXP52-b	EXP52-CPP. Do not rely on side effects in unevaluated operands
CERT_CPP-EXP52-c	EXP52-CPP. Do not rely on side effects in unevaluated operands

CERT_CPP-EXP53-a	EXP53-CPP. Do not read uninitialized memory
CERT_CPP-EXP54-a	EXP54-CPP. Do not access an object outside of its lifetime
CERT_CPP-EXP54-b	EXP54-CPP. Do not access an object outside of its lifetime
CERT_CPP-EXP54-c	EXP54-CPP. Do not access an object outside of its lifetime
CERT_CPP-EXP55-a	EXP55-CPP. Do not access a cv-qualified object through a cv-unqualified type
CERT_CPP-EXP56-a	EXP56-CPP. Do not call a function with a mismatched language linkage
CERT_CPP-EXP57-a	EXP57-CPP. Do not cast or delete pointers to incomplete classes
CERT_CPP-EXP57-b	EXP57-CPP. Do not cast or delete pointers to incomplete classes
CERT_CPP-EXP58-a	EXP58-CPP. Pass an object of the correct type to va_start
CERT_CPP-EXP59-a	EXP59-CPP. Use offsetof() on valid types and members
CERT_CPP-EXP60-a	EXP60-CPP. Do not pass a nonstandard-layout type object across execution boundaries
CERT_CPP-EXP61-a	EXP61-CPP. A lambda object must not outlive any of its reference captured objects
CERT_CPP-EXP61-b	EXP61-CPP. A lambda object must not outlive any of its reference captured objects
CERT_CPP-EXP61-c	EXP61-CPP. A lambda object must not outlive any of its reference captured objects
CERT_CPP-EXP62-a	EXP62-CPP. Do not access the bits of an object representation that are not part of the object's value representation
CERT_CPP-EXP63-a	EXP63-CPP. Do not rely on the value of a moved-from object
CERT_CPP-FIO50-a	FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
CERT_CPP-FIO51-a	FIO51-CPP. Close files when they are no longer needed
CERT_CPP-INT50-a	INT50-CPP. Do not cast to an out-of-range enumeration value
CERT_CPP-MEM50-a	MEM50-CPP. Do not access freed memory
CERT_CPP-MEM51-a	MEM51-CPP. Properly deallocate dynamically allocated resources
CERT_CPP-MEM51-b	MEM51-CPP. Properly deallocate dynamically allocated resources
CERT_CPP-MEM51-c	MEM51-CPP. Properly deallocate dynamically allocated resources
CERT_CPP-MEM51-d	MEM51-CPP. Properly deallocate dynamically allocated resources
CERT_CPP-MEM52-a	MEM52-CPP. Detect and handle memory allocation errors
CERT_CPP-MEM52-b	MEM52-CPP. Detect and handle memory allocation errors
CERT_CPP-MEM53-a	MEM53-CPP. Explicitly construct and destruct objects when manually managing object lifetime
CERT_CPP-MEM54-a	MEM54-CPP. Provide placement new with properly aligned pointers to sufficient storage capacity
CERT_CPP-MEM54-b	MEM54-CPP. Provide placement new with properly aligned pointers to sufficient storage capacity
CERT_CPP-MEM55-a	MEM55-CPP. Honor replacement dynamic storage management requirements
CERT_CPP-MEM56-a	MEM56-CPP. Do not store an already-owned pointer value in an unrelated smart pointer
CERT_CPP-MEM57-a	MEM57-CPP. Avoid using default operator new for over-aligned types
CERT_CPP-MSC50-a	MSC50-CPP. Do not use std::rand() for generating pseudorandom numbers
CERT_CPP-MSC51-a	MSC51-CPP. Ensure your random number generator is properly seeded
CERT_CPP-MSC52-a	MSC52-CPP. Value-returning functions must return a value from all exit paths
CERT_CPP-MSC53-a	MSC53-CPP. Do not return from a function declared [[noreturn]]
CERT_CPP-MSC54-a	MSC54-CPP. A signal handler must be a plain old function

CERT_CPP-OOP50-a	OOP50-CPP. Do not invoke virtual functions from constructors or destructors
CERT_CPP-OOP50-b	OOP50-CPP. Do not invoke virtual functions from constructors or destructors
CERT_CPP-OOP50-c	OOP50-CPP. Do not invoke virtual functions from constructors or destructors
CERT_CPP-OOP50-d	OOP50-CPP. Do not invoke virtual functions from constructors or destructors
CERT_CPP-OOP51-a	OOP51-CPP. Do not slice derived objects
CERT_CPP-OOP52-a	OOP52-CPP. Do not delete a polymorphic object without a virtual destructor
CERT_CPP-OOP53-a	OOP53-CPP. Write constructor member initializers in the canonical order
CERT_CPP-OOP54-a	OOP54-CPP. Gracefully handle self-copy assignment
CERT_CPP-OOP55-a	OOP55-CPP. Do not use pointer-to-member operators to access nonexistent members
CERT_CPP-OOP56-a	OOP56-CPP. Honor replacement handler requirements
CERT_CPP-OOP56-b	OOP56-CPP. Honor replacement handler requirements
CERT_CPP-OOP56-c	OOP56-CPP. Honor replacement handler requirements
CERT_CPP-OOP57-a	OOP57-CPP. Prefer special member functions and overloaded operators to C Standard Library functions
CERT_CPP-OOP57-b	OOP57-CPP. Prefer special member functions and overloaded operators to C Standard Library functions
CERT_CPP-OOP58-a	OOP58-CPP. Copy operations must not mutate the source object
CERT_CPP-STR50-a	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT_CPP-STR50-b	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT_CPP-STR50-c	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT_CPP-STR50-d	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT_CPP-STR50-e	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT_CPP-STR50-f	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT_CPP-STR50-g	STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
CERT_CPP-STR51-a	STR51-CPP. Do not attempt to create a std::string from a null pointer
CERT_CPP-STR52-a	STR52-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
CERT_CPP-STR53-a	STR53-CPP. Range check element access
Runtime detection	EXP51-CPP. Do not delete an array through a pointer of the incorrect type
Runtime detection	EXP53-CPP. Do not read uninitialized memory
Runtime detection	EXP54-CPP. Do not access an object outside of its lifetime
Runtime detection	EXP57-CPP. Do not cast or delete pointers to incomplete classes
Runtime detection	MEM50-CPP. Do not access freed memory
Runtime detection	MEM51-CPP. Properly deallocate dynamically allocated resources
Runtime detection	MEM52-CPP. Detect and handle memory allocation errors
Runtime detection	FIO51-CPP. Close files when they are no longer needed
Runtime detection	OOP55-CPP. Do not use pointer-to-member operators to access nonexistent members