GRAMMATECH

SEI CERT-C RULES AND RECOMMENDATIONS MAPPED TO CODESONAR® 7.1 WARNING CLASSES

TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

INTRODUCTION

The SEI CERT C Coding Standard (CERT-C) provides rules and recommendations for secure coding in the C programming language. The goal of these rules and recommendations is to develop safe, reliable, and secure systems, for example by eliminating undefined behaviors that can lead to undefined program behaviors and exploitable vulnerabilities. Conformance to the coding rules defined in this standard is necessary (but not sufficient) to ensure the safety, reliability, and security of software systems developed in the C programming language.

CodeSonar 7.1 includes a large number of warning classes that support checking for the CERT-C guidelines. Every CodeSonar warning report includes the numbers of any CERT-C rules and recommendations that are closely mapped to the warning's class. (The close mapping for a warning class is the set of categories—including CERT-C rules and recommendations—that most closely match the class, if any).

You can configure CodeSonar to enable and disable warning classes mapped to specific CERT-C rules and recommendations, or use build presets to enable all warning classes that are closely mapped to any CERT-C rules and recommendations. In addition, you can use the CodeSonar search function to find warnings related to specific CERT-C rules or recommendations, or to any CERT-C rule or recommendation.

For more information on the SEI CERT C Coding Standard:

https://www.securecoding.cert.org/confluence/display/c/

The remainder of this document comprises two tables:

- A table showing the close mapping between CodeSonar warning classes and the SEI CERTC Coding Standard.
- A table showing the broad mapping between CodeSonar warning classes and the SEI CERTC Coding Standard. The broad CERT-C mapping for a CodeSonar warning class includes the close CERT-C mapping for the class, plus any other CERT-C rules and recommendations thatare related to the class in a meaningful way, but not eligible for the close mapping.

GrammaTech is a leading global provider of application testing (AST) solutions used by the world's most security conscious organizations to detect, measure, analyze and resolve vulnerabilities for software they develop or use. The company is also a trusted cybersecurity and artificial intelligence research partner for the nation's civil, defense, and intelligence agencies.

SEI CERT C CODING STANDARD CLOSE MAPPING (CODESONAR V7.1)

The following table contains CodeSonar warning classes that are closely mapped to CERT-C rules and recommendations.

Rule	Rule Name	Supported
CERT-C:API00-C	Functions should validate their parameters	Yes
CERT-C:API01-C	Avoid laying out strings in memory directly before sensitive data	No
CERT-C:API02-C	Functions that read or write to or from an array should take an argument to specify the source or target size	No
CERT-C:API03-C	Create consistent interfaces and capabilities across related functions	No
CERT-C:API04-C	Provide a consistent and usable error-checking mechanism	No
CERT-C:API05-C	Use conformant array parameters	No
CERT-C:API07-C	Enforce type safety	Yes
CERT-C:API08-C	Avoid parameter names in a function prototype	No
CERT-C:API09-C	Compatible values should have the same type	No
CERT-C:API10-C	APIs should have security options enabled by default	No
CERT-C:ARR00-C	Understand how arrays work	No
CERT-C:ARR01-C	Do not apply the sizeof operator to a pointer when taking the size of an array	No
CERT-C:ARR02-C	Explicitly specify array bounds, even if implicitly defined by an initializer	No
CERT-C:ARR30-C	Do not form or use out-of-bounds pointers or array subscripts	Yes
CERT-C:ARR32-C	Ensure size arguments for variable length arrays are in a valid range	Yes
CERT-C:ARR36-C	Do not subtract or compare two pointers that do not refer to the same array	Yes
CERT-C:ARR37-C	Do not add or subtract an integer to a pointer to a non-array object	Yes
CERT-C:ARR38-C	Guarantee that library functions do not form invalid pointers	Yes
CERT-C:ARR39-C	Do not add or subtract a scaled integer to a pointer	Yes
CERT-C:CON01-C	Acquire and release synchronization primitives in the same module, at the same level of abstraction	Yes
CERT-C:CON02-C	Do not use volatile as a synchronization primitive	No
CERT-C:CON03-C	Ensure visibility when accessing shared variables	No
CERT-C:CON04-C	Join or detach threads even if their exit status is unimportant	No
CERT-C:CON05-C	Do not perform operations that can block while holding a lock	Yes
CERT-C:CON06-C	Ensure that every mutex outlives the data it protects	No
CERT-C:CON07-C	Ensure that compound operations on shared variables are atomic	Yes
CERT-C:CON08-C	Do not assume that a group of calls to independently atomic methods is atomic	No
CERT-C:CON09-C	Avoid the ABA problem when using lock-free algorithms	No
CERT-C:CON30-C	Clean up thread-specific storage	Yes
CERT-C:CON31-C	Do not destroy a mutex while it is locked	Yes
CERT-C:CON32-C	Prevent data races when accessing bit-fields from multiple threads	Yes
CERT-C:CON33-C	Avoid race conditions when using library functions	Yes
CERT-C:CON34-C	Declare objects shared between threads with appropriate storage durations	Yes
CERT-C:CON35-C	Avoid deadlock by locking in a predefined order	Yes
CERT-C:CON36-C	Wrap functions that can spuriously wake up in a loop	Yes
CERT-C:CON37-C	Do not call signal() in a multithreaded program	Yes
CERT-C:CON38-C	Preserve thread safety and liveness when using condition variables	Yes
CERT-C:CON39-C	Do not join or detach a thread that was previously joined or detached	Yes
CERT-C:CON40-C	Do not refer to an atomic variable twice in an expression	Yes
CERT-C:CON41-C	Wrap functions that can fail spuriously in a loop	Yes
CERT-C:CON43-C	Do not allow data races in multithreaded code	Yes
CERT-C:DCL00-C	Const-qualify immutable objects	Yes

CERT-C:DCL01-C	Do not reuse variable names in subscopes	Yes
CERT-C:DCL02-C	Use visually distinct identifiers	Yes
CERT-C:DCL03-C	Use a static assertion to test the value of a constant expression	No
CERT-C:DCL04-C	Do not declare more than one variable per declaration	Yes
CERT-C:DCL05-C	Use typedefs of non-pointer types only	Yes
CERT-C:DCL06-C	Use meaningful symbolic constants to represent literal values	No
CERT-C:DCL07-C	Include the appropriate type information in function declarators	Yes
CERT-C:DCL08-C	Properly encode relationships in constant definitions	No
CERT-C:DCL09-C	Declare functions that return errno with a return type of errno_t	No
CERT-C:DCL10-C	Maintain the contract between the writer and caller of variadic functions	No
CERT-C:DCL11-C	Understand the type issues associated with variadic functions	Yes
CERT-C:DCL12-C	Implement abstract data types using opaque types	No
CERT-C:DCL13-C	Declare function parameters that are pointers to values not changed by the function as const	Yes
CERT-C:DCL15-C	Declare file-scope objects or functions that do not need external linkage as static	Yes
CERT-C:DCL16-C	Use "L," not "l," to indicate a long value	Yes
CERT-C:DCL17-C	Beware of miscompiled volatile-qualified variables	No
CERT-C:DCL18-C	Do not begin integer constants with 0 when specifying a decimal value	Yes
CERT-C:DCL19-C	Minimize the scope of variables and functions	Yes
CERT-C:DCL20-C	Explicitly specify void when a function accepts no arguments	Yes
CERT-C:DCL21-C	Understand the storage of compound literals	No
CERT-C:DCL22-C	Use volatile for data that cannot be cached	No
CERT-C:DCL23-C	Guarantee that mutually visible identifiers are unique	Yes
CERT-C:DCL30-C	Declare objects with appropriate storage durations	Yes
CERT-C:DCL31-C	Declare identifiers before using them	No
CERT-C:DCL36-C	Do not declare an identifier with conflicting linkage classifications	Yes
CERT-C:DCL37-C	Do not declare or define a reserved identifier	Yes
CERT-C:DCL38-C	Use the correct syntax when declaring a flexible array member	No
CERT-C:DCL39-C	Avoid information leakage when passing a structure across a trust boundary	Yes
CERT-C:DCL40-C	Do not create incompatible declarations of the same function or object	Yes
CERT-C:DCL41-C	Do not declare variables inside a switch statement before the first case label	Yes
CERT-C:ENV01-C	Do not make assumptions about the size of an environment variable	Yes
CERT-C:ENV02-C	Beware of multiple environment variables with the same effective name	No
CERT-C:ENV03-C	Sanitize the environment when invoking external programs	No
CERT-C:ENV30-C	Do not modify the object referenced by the return value of certain functions	Yes
CERT-C:ENV31-C	Do not rely on an environment pointer following an operation that may invalidate it	No
CERT-C:ENV32-C	All exit handlers must return normally	Yes
CERT-C:ENV33-C	Do not call system()	Yes
CERT-C:ENV34-C	Do not store pointers returned by certain functions	No
CERT-C:ERR00-C	Adopt and implement a consistent and comprehensive error-handling policy	No
CERT-C:ERR01-C	Use ferror() rather than errno to check for FILE stream errors	No
CERT-C:ERR02-C	Avoid in-band error indicators	No
CERT-C:ERR03-C	Use runtime-constraint handlers when calling the bounds-checking interfaces	No
CERT-C:ERR04-C	Choose an appropriate termination strategy	No
CERT-C:ERR05-C	Application-independent code should provide error detection without dictating error handling	No
CERT-C:ERR06-C	Understand the termination behavior of assert() and abort()	No
CERT-C:ERR07-C	Prefer functions that support error checking over equivalent functions that don't	No
CERT-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	Yes
CERT-C:ERR32-C	Do not rely on indeterminate values of errno	No

CERT-C:ERR33-C	Detect and handle standard library errors	Yes
CERT-C:ERR34-C	Detect errors when converting a string to a number	Yes
CERT-C:EXP00-C	Use parentheses for precedence of operation	Yes
CERT-C:EXP02-C	Be aware of the short-circuit behavior of the logical AND and OR operators	No
CERT-C:EXP03-C	Do not assume the size of a structure is the sum of the sizes of its members	No
CERT-C:EXP05-C	Do not cast away a const qualification	Yes
CERT-C:EXP07-C	Do not diminish the benefits of constants by assuming their values in expressions	No
CERT-C:EXP08-C	Ensure pointer arithmetic is used correctly	Yes
CERT-C:EXP09-C	Use sizeof to determine the size of a type or variable	No
CERT-C:EXP10-C	Do not depend on the order of evaluation of subexpressions or the order in which side effects take place	No
CERT-C:EXP11-C	Do not make assumptions regarding the layout of structures with bit-fields	No
CERT-C:EXP12-C	Do not ignore values returned by functions	Yes
CERT-C:EXP13-C	Treat relational and equality operators as if they were nonassociative	No
CERT-C:EXP14-C	Beware of integer promotion when performing bitwise operations on integer types smaller than int	Yes
CERT-C:EXP15-C	Do not place a semicolon on the same line as an if, for, or while statement	Yes
CERT-C:EXP16-C	Do not compare function pointers to constant values	No
CERT-C:EXP19-C	Use braces for the body of an if, for, or while statement	No
CERT-C:EXP20-C	Perform explicit tests to determine success, true and false, and equality	No
CERT-C:EXP30-C	Do not depend on the order of evaluation for side effects	Yes
CERT-C:EXP32-C	Do not access a volatile object through a nonvolatile reference	No
CERT-C:EXP33-C	Do not read uninitialized memory	Yes
CERT-C:EXP34-C	Do not dereference null pointers	Yes
CERT-C:EXP35-C	Do not modify objects with temporary lifetime	No
CERT-C:EXP36-C	Do not cast pointers into more strictly aligned pointer types	Yes
CERT-C:EXP37-C	Call functions with the correct number and type of arguments	Yes
CERT-C:EXP39-C	Do not access a variable through a pointer of an incompatible type	No
CERT-C:EXP40-C	Do not modify constant objects	No
CERT-C:EXP42-C	Do not compare padding data	Yes
CERT-C:EXP43-C	Avoid undefined behavior when using restrict-qualified pointers	Yes
CERT-C:EXP44-C	Do not rely on side effects in operands to sizeof, _Alignof, or _Generic	Yes
CERT-C:EXP45-C	Do not perform assignments in selection statements	Yes
CERT-C:EXP46-C	Do not use a bitwise operator with a Boolean-like operand	Yes
CERT-C:EXP47-C	Do not call va_arg with an argument of the incorrect type	Yes
CERT-C:FIO01-C	Be careful using functions that use file names for identification	Yes
CERT-C:FIO02-C	Canonicalize path names originating from tainted sources	Yes
CERT-C:FIO03-C	Do not make assumptions about fopen() and file creation	No
CERT-C:FIO05-C	Identify files using multiple file attributes	No
CERT-C:FIO06-C	Create files with appropriate access permissions	Yes
CERT-C:FIO08-C	Take care when calling remove() on an open file	No
CERT-C:FIO09-C	Be careful with binary data when transferring data across systems	No
CERT-C:FIO10-C	Take care when using the rename() function	No
CERT-C:FIO11-C	Take care when specifying the mode parameter of fopen()	No
CERT-C:FIO13-C	Never push back anything other than one read character	No
CERT-C:FIO14-C	Understand the difference between text mode and binary mode with file streams	No
CERT-C:FIO15-C	Ensure that file operations are performed in a secure directory	No
CERT-C:FIO17-C	Do not rely on an ending null character when using fread()	No
CERT-C:FIO18-C	Never expect fwrite() to terminate the writing process at a null character	No
CERT-C:FIO19-C	Do not use fseek() and ftell() to compute the size of a regular file	No
CERT-C:FIO20-C	Avoid unintentional truncation when using fgets() or fgetws()	No

CERT-C:FIO21-C	Do not create temporary files in shared directories	Yes
CERT-C:FIO22-C	Close files before spawning processes	No
CERT-C:FIO23-C	Do not exit with unflushed data in stdout or stderr	No
CERT-C:FIO24-C	Do not open a file that is already open	Yes
CERT-C:FIO30-C	Exclude user input from format strings	Yes
CERT-C:FIO32-C	Do not perform operations on devices that are only appropriate for files	No
CERT-C:FIO34-C	Distinguish between characters read from a file and EOF or WEOF	Yes
CERT-C:FIO37-C	Do not assume that fgets() or fgetws() returns a nonempty string when successful	Yes
CERT-C:FIO38-C	Do not copy a FILE object	No
CERT-C:FIO39-C	Do not alternately input and output from a stream without an intervening flush or positioning call	Yes
CERT-C:FIO40-C	Reset strings on fgets() or fgetws() failure	Yes
CERT-C:FIO41-C	Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects	No
CERT-C:FIO42-C	Close files when they are no longer needed	Yes
CERT-C:FIO44-C	Only use values for fsetpos() that are returned from fgetpos()	No
CERT-C:FIO45-C	Avoid TOCTOU race conditions while accessing files	Yes
CERT-C:FIO46-C	Do not access a closed file	Yes
CERT-C:FIO47-C	Use valid format strings	Yes
CERT-C:FLP00-C	Understand the limitations of floating-point numbers	No
CERT-C:FLP01-C	Take care in rearranging floating-point expressions	No
CERT-C:FLP02-C	Avoid using floating-point numbers when precise computation is needed	No
CERT-C:FLP03-C	Detect and handle floating-point errors	No
CERT-C:FLP04-C	Check floating-point inputs for exceptional values	No
CERT-C:FLP05-C	Do not use denormalized numbers	No
CERT-C:FLP06-C	Convert integers to floating point for floating-point operations	Yes
CERT-C:FLP07-C	Cast the return value of a function that returns a floating-point type	No
CERT-C:FLP30-C	Do not use floating-point variables as loop counters	Yes
CERT-C:FLP32-C	Prevent or detect domain and range errors in math functions	Yes
CERT-C:FLP34-C	Ensure that floating-point conversions are within range of the new type	Yes
CERT-C:FLP36-C	Preserve precision when converting integral values to floating-point type	Yes
CERT-C:FLP37-C	Do not use object representations to compare floating-point values	No
CERT-C:INT00-C	Understand the data model used by your implementation(s)	No
CERT-C:INT01-C	Use rsize_t or size_t for all integer values representing the size of an object	No
CERT-C:INT02-C	Understand integer conversion rules	Yes
CERT-C:INT04-C	Enforce limits on integer values originating from tainted sources	Yes
CERT-C:INT05-C	Do not use input functions to convert character data if they cannot handle all possible inputs	Yes
CERT-C:INT07-C	Use only explicitly signed or unsigned char type for numeric values	Yes
CERT-C:INT08-C	Verify that all integer values are in range	Yes
CERT-C:INT09-C	Ensure enumeration constants map to unique values	Yes
CERT-C:INT10-C	Do not assume a positive remainder when using the % operator	No
CERT-C:INT12-C	Do not make assumptions about the type of a plain int bit-field when used in an expression	Yes
CERT-C:INT13-C	Use bitwise operators only on unsigned operands	Yes
CERT-C:INT14-C	Avoid performing bitwise and arithmetic operations on the same data	No
CERT-C:INT15-C	Use intmax_t or uintmax_t for formatted IO on programmer-defined integer types	No
CERT-C:INT16-C	Do not make assumptions about representation of signed integers	No
CERT-C:INT17-C	Define integer constants in an implementation-independent manner	No
CERT-C:INT18-C	Evaluate integer expressions in a larger size before comparing or assigning to that size	Yes
CERT-C:INT30-C	Ensure that unsigned integer operations do not wrap	Yes
CERT-C:INT31-C	Ensure that integer conversions do not result in lost or misinterpreted data	Yes
CERT-C:INT32-C	Ensure that operations on signed integers do not result in overflow	Yes

CERT-C:INT33-C	Ensure that division and remainder operations do not result in divide-by-zero errors	Yes
CERT-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	Yes
CERT-C:INT35-C	Use correct integer precisions	Yes
CERT-C:INT36-C	Converting a pointer to integer or integer to pointer	Yes
CERT-C:MEM00-C	Allocate and free memory in the same module, at the same level of abstraction	Yes
CERT-C:MEM01-C	Store a new value in pointers immediately after free()	Yes
CERT-C:MEM02-C	Immediately cast the result of a memory allocation function call into a pointer to the allocated type	No
CERT-C:MEM03-C	Clear sensitive information stored in reusable resources	No
CERT-C:MEM04-C	Beware of zero-length allocations	No
CERT-C:MEM05-C	Avoid large stack allocations	Yes
CERT-C:MEM06-C	Ensure that sensitive data is not written out to disk	No
CERT-C:MEM07-C	Ensure that the arguments to calloc(), when multiplied, do not wrap	Yes
CERT-C:MEM10-C	Define and use a pointer validation function	No
CERT-C:MEM11-C	Do not assume infinite heap space	Yes
CERT-C:MEM12-C	Consider using a goto chain when leaving a function on error when using and releasing resources	No
CERT-C:MEM30-C	Do not access freed memory	Yes
CERT-C:MEM31-C	Free dynamically allocated memory when no longer needed	Yes
CERT-C:MEM33-C	Allocate and copy structures containing a flexible array member dynamically	Yes
CERT-C:MEM34-C	Only free memory allocated dynamically	Yes
CERT-C:MEM35-C	Allocate sufficient memory for an object	Yes
CERT-C:MEM36-C	Do not modify the alignment of objects by calling realloc()	Yes
CERT-C:MSC00-C	Compile cleanly at high warning levels	Yes
CERT-C:MSC01-C	Strive for logical completeness	No
CERT-C:MSC04-C	Use comments consistently and in a readable fashion	No
CERT-C:MSC05-C	Do not manipulate time_t typed values directly	No
CERT-C:MSC06-C	Beware of compiler optimizations	Yes
CERT-C:MSC07-C	Detect and remove dead code	Yes
CERT-C:MSC09-C	Character encoding: Use subset of ASCII for safety	No
CERT-C:MSC10-C	Character encoding: UTF8-related issues	No
CERT-C:MSC11-C	Incorporate diagnostic tests using assertions	Yes
CERT-C:MSC12-C	Detect and remove code that has no effect or is never executed	Yes
CERT-C:MSC13-C	Detect and remove unused values	Yes
CERT-C:MSC14-C	Do not introduce unnecessary platform dependencies	No
CERT-C:MSC15-C	Do not depend on undefined behavior	No
CERT-C:MSC17-C	Finish every set of statements associated with a case label with a break statement	Yes
CERT-C:MSC18-C	Be careful while handling sensitive data, such as passwords, in program code	Yes
CERT-C:MSC19-C	For functions that return an array, prefer returning an empty array over a null value	No
CERT-C:MSC20-C	Do not use a switch statement to transfer control into a complex block	Yes
CERT-C:MSC21-C	Use robust loop termination conditions	Yes
CERT-C:MSC22-C	Use the setjmp(), longjmp() facility securely	Yes
CERT-C:MSC23-C	Beware of vendor-specific library and language differences	No
CERT-C:MSC24-C	Do not use deprecated or obsolescent functions	Yes
CERT-C:MSC30-C	Do not use the rand() function for generating pseudorandom numbers	Yes
CERT-C:MSC32-C	Properly seed pseudorandom number generators	Yes
CERT-C:MSC33-C	Do not pass invalid data to the asctime() function	Yes
CERT-C:MSC37-C	Ensure that control never reaches the end of a non-void function	Yes
CERT-C:MSC38-C	Do not treat a predefined identifier as an object if it might only be implemented as a macro	Yes
CERT-C:MSC39-C	Do not call va_arg() on a va_list that has an indeterminate value	Yes

CERT-C:MSC40-C	Do not violate constraints	No
CERT-C:MSC41-C	Never hard code sensitive information	Yes
CERT-C:POS01-C	Check for the existence of links when dealing with files	No
CERT-C:POS02-C	Follow the principle of least privilege	No
CERT-C:POS04-C	Avoid using PTHREAD_MUTEX_NORMAL type mutex locks	No
CERT-C:POS05-C	Limit access to files by creating a jail	Yes
CERT-C:POS30-C	Use the readlink() function properly	Yes
CERT-C:POS33-C	Do not use vfork()	Yes
CERT-C:POS34-C	Do not call putenv() with a pointer to an automatic variable as the argument	Yes
CERT-C:POS35-C	Avoid race conditions while checking for the existence of a symbolic link	No
CERT-C:POS36-C	Observe correct revocation order while relinquishing privileges	No
CERT-C:POS37-C	Ensure that privilege relinquishment is successful	No
CERT-C:POS38-C	Beware of race conditions when using fork and file descriptors	Yes
CERT-C:POS39-C	Use the correct byte ordering when transferring data between systems	No
CERT-C:POS44-C	Do not use signals to terminate threads	Yes
CERT-C:POS47-C	Do not use threads that can be canceled asynchronously	No
CERT-C:POS48-C	Do not unlock or destroy another POSIX thread's mutex	Yes
CERT-C:POS49-C	When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed	Yes
CERT-C:POS50-C	Declare objects shared between POSIX threads with appropriate storage durations	No
CERT-C:POS51-C	Avoid deadlock with POSIX threads by locking in predefined order	Yes
CERT-C:POS52-C	Do not perform operations that can block while holding a POSIX lock	Yes
CERT-C:POS53-C	Do not use more than one mutex for concurrent waiting operations on a condition variable	No
CERT-C:POS54-C	Detect and handle POSIX library errors	Yes
CERT-C:PRE00-C	Prefer inline or static functions to function-like macros	Yes
CERT-C:PRE01-C	Use parentheses within macros around parameter names	No
CERT-C:PRE02-C	Macro replacement lists should be parenthesized	Yes
CERT-C:PRE03-C	Prefer typedefs to defines for encoding non-pointer types	No
CERT-C:PRE04-C	Do not reuse a standard header file name	No
CERT-C:PRE05-C	Understand macro replacement when concatenating tokens or performing stringification	Yes
CERT-C:PRE06-C	Enclose header files in an include guard	No
CERT-C:PRE07-C	Avoid using repeated question marks	No
CERT-C:PRE08-C	Guarantee that header file names are unique	No
CERT-C:PRE09-C	Do not replace secure functions with deprecated or obsolescent functions	No
CERT-C:PRE10-C	Wrap multistatement macros in a do-while loop	No
CERT-C:PRE11-C	Do not conclude macro definitions with a semicolon	Yes
CERT-C:PRE12-C	Do not define unsafe macros	No
CERT-C:PRE13-C	Use the Standard predefined macros to test for versions and features.	No
CERT-C:PRE30-C	Do not create a universal character name through concatenation	Yes
CERT-C:PRE31-C	Avoid side effects in arguments to unsafe macros	Yes
CERT-C:PRE32-C	Do not use preprocessor directives in invocations of function-like macros	Yes
CERT-C:SIG00-C	Mask signals handled by noninterruptible signal handlers	Yes
CERT-C:SIG01-C	Understand implementation-specific details regarding signal handler persistence	Yes
CERT-C:SIG02-C	Avoid using signals to implement normal functionality	Yes
CERT-C:SIG30-C	Call only asynchronous-safe functions within signal handlers	Yes
CERT-C:SIG31-C	Do not access shared objects in signal handlers	Yes
CERT-C:SIG34-C	Do not call signal() from within interruptible signal handlers	Yes
CERT-C:SIG35-C	Do not return from a computational exception signal handler	Yes
CERT-C:STR00-C	Represent characters using an appropriate type	Yes
CERT-C:STR01-C	Adopt and implement a consistent plan for managing strings	No

CERT-C:STR02-C	Sanitize data passed to complex subsystems	Yes
CERT-C:STR03-C	Do not inadvertently truncate a string	Yes
CERT-C:STR04-C	Use plain char for characters in the basic character set	Yes
CERT-C:STR05-C	Use pointers to const when referring to string literals	Yes
CERT-C:STR06-C	Do not assume that strtok() leaves the parse string unchanged	No
CERT-C:STR07-C	Use the bounds-checking interfaces for string manipulation	Yes
CERT-C:STR08-C	Use managed strings for development of new string manipulation code	No
CERT-C:STR09-C	Don't assume numeric values for expressions with type plain character	No
CERT-C:STR10-C	Do not concatenate different type of string literals	No
CERT-C:STR11-C	Do not specify the bound of a character array initialized with a string literal	No
CERT-C:STR30-C	Do not attempt to modify string literals	No
CERT-C:STR31-C	Guarantee that storage for strings has sufficient space for character data and the null terminator	Yes
CERT-C:STR32-C	Do not pass a non-null-terminated character sequence to a library function that expects a string	Yes
CERT-C:STR34-C	Cast characters to unsigned char before converting to larger integer sizes	Yes
CERT-C:STR37-C	Arguments to character-handling functions must be representable as an unsigned char	Yes
CERT-C:STR38-C	Do not confuse narrow and wide character strings and functions	Yes
CERT-C:WIN00-C	Be specific when dynamically loading libraries	Yes
CERT-C:WIN01-C	Do not forcibly terminate execution	No
CERT-C:WIN02-C	Restrict privileges when spawning child processes	Yes
CERT-C:WIN03-C	Understand HANDLE inheritance	No
CERT-C:WIN04-C	Consider encrypting function pointers	No
CERT-C:WIN30-C	Properly pair allocation and deallocation functions	Yes

SEI CERT C CODING STANDARD BROAD MAPPING (CODESONAR V7.1)

The following table contains CodeSonar warning classes that are broadly mapped to CERT-C rulesand recommendations.

Rule	Rule Name	Supported
CERT-C:API00-C	Functions should validate their parameters	Yes
CERT-C:API01-C	Avoid laying out strings in memory directly before sensitive data	No
CERT-C:API02-C	Functions that read or write to or from an array should take an argument to specify the source or target size	No
CERT-C:API03-C	Create consistent interfaces and capabilities across related functions	No
CERT-C:API04-C	Provide a consistent and usable error-checking mechanism	No
CERT-C:API05-C	Use conformant array parameters	No
CERT-C:API07-C	Enforce type safety	Yes
CERT-C:API08-C	Avoid parameter names in a function prototype	No
CERT-C:API09-C	Compatible values should have the same type	No
CERT-C:API10-C	APIs should have security options enabled by default	No
CERT-C:ARR00-C	Understand how arrays work	No
CERT-C:ARR01-C	Do not apply the sizeof operator to a pointer when taking the size of an array	No
CERT-C:ARR02-C	Explicitly specify array bounds, even if implicitly defined by an initializer	No
CERT-C:ARR30-C	Do not form or use out-of-bounds pointers or array subscripts	Yes
CERT-C:ARR32-C	Ensure size arguments for variable length arrays are in a valid range	Yes
CERT-C:ARR36-C	Do not subtract or compare two pointers that do not refer to the same array	Yes
CERT-C:ARR37-C	Do not add or subtract an integer to a pointer to a non-array object	Yes
CERT-C:ARR38-C	Guarantee that library functions do not form invalid pointers	Yes
CERT-C:ARR39-C	Do not add or subtract a scaled integer to a pointer	Yes
CERT-C:CON01-C	Acquire and release synchronization primitives in the same module, at the same level of abstraction	Yes
CERT-C:CON02-C	Do not use volatile as a synchronization primitive	No
CERT-C:CON03-C	Ensure visibility when accessing shared variables	No
CERT-C:CON04-C	Join or detach threads even if their exit status is unimportant	No
CERT-C:CON05-C	Do not perform operations that can block while holding a lock	Yes
CERT-C:CON06-C	Ensure that every mutex outlives the data it protects	Yes
CERT-C:CON07-C	Ensure that compound operations on shared variables are atomic	Yes
CERT-C:CON08-C	Do not assume that a group of calls to independently atomic methods is atomic	No
CERT-C:CON09-C	Avoid the ABA problem when using lock-free algorithms	No
CERT-C:CON30-C	Clean up thread-specific storage	Yes
CERT-C:CON31-C	Do not destroy a mutex while it is locked	Yes
CERT-C:CON32-C	Prevent data races when accessing bit-fields from multiple threads	Yes
CERT-C:CON33-C	Avoid race conditions when using library functions	Yes
CERT-C:CON34-C	Declare objects shared between threads with appropriate storage durations	Yes
CERT-C:CON35-C	Avoid deadlock by locking in a predefined order	Yes
CERT-C:CON36-C	Wrap functions that can spuriously wake up in a loop	Yes
CERT-C:CON37-C	Do not call signal() in a multithreaded program	Yes
CERT-C:CON38-C	Preserve thread safety and liveness when using condition variables	Yes
CERT-C:CON39-C	Do not join or detach a thread that was previously joined or detached	Yes
CERT-C:CON40-C	Do not refer to an atomic variable twice in an expression	Yes
CERT-C:CON41-C	Wrap functions that can fail spuriously in a loop	Yes

CERT-C:CON43-C	Do not allow data races in multithreaded code	Yes
CERT-C:DCL00-C	Const-qualify immutable objects	Yes
CERT-C:DCL01-C	Do not reuse variable names in subscopes	Yes
CERT-C:DCL02-C	Use visually distinct identifiers	Yes
CERT-C:DCL03-C	Use a static assertion to test the value of a constant expression	No
CERT-C:DCL04-C	Do not declare more than one variable per declaration	Yes
CERT-C:DCL05-C	Use typedefs of non-pointer types only	Yes
CERT-C:DCL06-C	Use meaningful symbolic constants to represent literal values	No
CERT-C:DCL07-C	Include the appropriate type information in function declarators	Yes
CERT-C:DCL08-C	Properly encode relationships in constant definitions	No
CERT-C:DCL09-C	Declare functions that return errno with a return type of errno_t	No
CERT-C:DCL10-C	Maintain the contract between the writer and caller of variadic functions	No
CERT-C:DCL11-C	Understand the type issues associated with variadic functions	Yes
CERT-C:DCL12-C	Implement abstract data types using opaque types	No
CERT-C:DCL13-C	Declare function parameters that are pointers to values not changed by the function as const	Yes
CERT-C:DCL15-C	Declare file-scope objects or functions that do not need external linkage as static	Yes
CERT-C:DCL16-C	Use "L," not "l," to indicate a long value	Yes
CERT-C:DCL17-C	Beware of miscompiled volatile-qualified variables	No
CERT-C:DCL18-C	Do not begin integer constants with 0 when specifying a decimal value	Yes
CERT-C:DCL19-C	Minimize the scope of variables and functions	Yes
CERT-C:DCL20-C	Explicitly specify void when a function accepts no arguments	Yes
CERT-C:DCL21-C	Understand the storage of compound literals	No
CERT-C:DCL22-C	Use volatile for data that cannot be cached	No
CERT-C:DCL23-C	Guarantee that mutually visible identifiers are unique	Yes
CERT-C:DCL30-C	Declare objects with appropriate storage durations	Yes
CERT-C:DCL31-C	Declare identifiers before using them	No
CERT-C:DCL36-C	Do not declare an identifier with conflicting linkage classifications	Yes
CERT-C:DCL37-C	Do not declare or define a reserved identifier	Yes
CERT-C:DCL38-C	Use the correct syntax when declaring a flexible array member	Yes
CERT-C:DCL39-C	Avoid information leakage when passing a structure across a trust boundary	Yes
CERT-C:DCL40-C	Do not create incompatible declarations of the same function or object	Yes
CERT-C:DCL41-C	Do not declare variables inside a switch statement before the first case label	Yes
CERT-C:ENV01-C	Do not make assumptions about the size of an environment variable	Yes
CERT-C:ENV02-C	Beware of multiple environment variables with the same effective name	No
CERT-C:ENV03-C	Sanitize the environment when invoking external programs	No
CERT-C:ENV30-C	Do not modify the object referenced by the return value of certain functions	Yes
CERT-C:ENV31-C	Do not rely on an environment pointer following an operation that may invalidate it	Yes
CERT-C:ENV32-C	All exit handlers must return normally	Yes
CERT-C:ENV33-C	Do not call system()	Yes
CERT-C:ENV34-C	Do not store pointers returned by certain functions	Yes
CERT-C:ERR00-C	Adopt and implement a consistent and comprehensive error-handling policy	No
CERT-C:ERR01-C	Use ferror() rather than errno to check for FILE stream errors	No
CERT-C:ERR02-C	Avoid in-band error indicators	No
CERT-C:ERR03-C	Use runtime-constraint handlers when calling the bounds-checking interfaces	No
CERT-C:ERR04-C	Choose an appropriate termination strategy	No
CERT-C:ERR05-C	Application-independent code should provide error detection without dictating error handling	No
CERT-C:ERR06-C	Understand the termination behavior of assert() and abort()	No
CERT-C:ERR07-C	Prefer functions that support error checking over equivalent functions that don't	No

CERT-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	Yes
CERT-C:ERR32-C	Do not rely on indeterminate values of errno	No
CERT-C:ERR33-C	Detect and handle standard library errors	Yes
CERT-C:ERR34-C	Detect errors when converting a string to a number	Yes
CERT-C:EXP00-C	Use parentheses for precedence of operation	Yes
CERT-C:EXP02-C	Be aware of the short-circuit behavior of the logical AND and OR operators	No
CERT-C:EXP03-C	Do not assume the size of a structure is the sum of the sizes of its members	No
CERT-C:EXP05-C	Do not cast away a const qualification	Yes
CERT-C:EXP07-C	Do not diminish the benefits of constants by assuming their values in expressions	No
CERT-C:EXP08-C	Ensure pointer arithmetic is used correctly	Yes
CERT-C:EXP09-C	Use sizeof to determine the size of a type or variable	No
CERT-C:EXP10-C	Do not depend on the order of evaluation of subexpressions or the order in which side effects take place	No
CERT-C:EXP11-C	Do not make assumptions regarding the layout of structures with bit-fields	No
CERT-C:EXP12-C	Do not ignore values returned by functions	Yes
CERT-C:EXP13-C	Treat relational and equality operators as if they were nonassociative	No
CERT-C:EXP14-C	Beware of integer promotion when performing bitwise operations on integer types smaller than int	Yes
CERT-C:EXP15-C	Do not place a semicolon on the same line as an if, for, or while statement	Yes
CERT-C:EXP16-C	Do not compare function pointers to constant values	No
CERT-C:EXP19-C	Use braces for the body of an if, for, or while statement	No
CERT-C:EXP20-C	Perform explicit tests to determine success, true and false, and equality	No
CERT-C:EXP30-C	Do not depend on the order of evaluation for side effects	Yes
CERT-C:EXP32-C	Do not access a volatile object through a nonvolatile reference	No
CERT-C:EXP33-C	Do not read uninitialized memory	Yes
CERT-C:EXP34-C	Do not dereference null pointers	Yes
CERT-C:EXP35-C	Do not modify objects with temporary lifetime	No
CERT-C:EXP36-C	Do not cast pointers into more strictly aligned pointer types	Yes
CERT-C:EXP37-C	Call functions with the correct number and type of arguments	Yes
CERT-C:EXP39-C	Do not access a variable through a pointer of an incompatible type	Yes
CERT-C:EXP40-C	Do not modify constant objects	Yes
CERT-C:EXP42-C	Do not compare padding data	Yes
CERT-C:EXP43-C	Avoid undefined behavior when using restrict-qualified pointers	Yes
CERT-C:EXP44-C	Do not rely on side effects in operands to sizeof, _Alignof, or _Generic	Yes
CERT-C:EXP45-C	Do not perform assignments in selection statements	Yes
CERT-C:EXP46-C	Do not use a bitwise operator with a Boolean-like operand	Yes
CERT-C:EXP47-C	Do not call va_arg with an argument of the incorrect type	Yes
CERT-C:FIO01-C	Be careful using functions that use file names for identification	Yes
CERT-C:FIO02-C	Canonicalize path names originating from tainted sources	Yes
CERT-C:FIO03-C	Do not make assumptions about fopen() and file creation	No
CERT-C:FIO05-C	Identify files using multiple file attributes	No
CERT-C:FIO06-C	Create files with appropriate access permissions	Yes
CERT-C:FIO08-C	Take care when calling remove() on an open file	No
CERT-C:FIO09-C	Be careful with binary data when transferring data across systems	No
CERT-C:FIO10-C	Take care when using the rename() function	No
CERT-C:FIO11-C	Take care when specifying the mode parameter of fopen()	No
CERT-C:FIO13-C	Never push back anything other than one read character	No
CERT-C:FIO14-C	Understand the difference between text mode and binary mode with file streams	No
CERT-C:FIO15-C	Ensure that file operations are performed in a secure directory	No

CERT-C:FIO17-C	Do not rely on an ending null character when using fread()	No
CERT-C:FIO18-C	Never expect fwrite() to terminate the writing process at a null character	No
CERT-C:FIO19-C	Do not use fseek() and ftell() to compute the size of a regular file	No
CERT-C:FIO20-C	Avoid unintentional truncation when using fgets() or fgetws()	No
CERT-C:FIO21-C	Do not create temporary files in shared directories	Yes
CERT-C:FIO22-C	Close files before spawning processes	No
CERT-C:FIO23-C	Do not exit with unflushed data in stdout or stderr	No
CERT-C:FIO24-C	Do not open a file that is already open	Yes
CERT-C:FIO30-C	Exclude user input from format strings	Yes
CERT-C:FIO32-C	Do not perform operations on devices that are only appropriate for files	No
CERT-C:FIO34-C	Distinguish between characters read from a file and EOF or WEOF	Yes
CERT-C:FIO37-C	Do not assume that fgets() or fgetws() returns a nonempty string when successful	Yes
CERT-C:FIO38-C	Do not copy a FILE object	No
CERT-C:FIO39-C	Do not alternately input and output from a stream without an intervening flush or positioning call	Yes
CERT-C:FIO40-C	Reset strings on fgets() or fgetws() failure	Yes
CERT-C:FIO41-C	Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects	Yes
CERT-C:FIO42-C	Close files when they are no longer needed	Yes
CERT-C:FIO44-C	Only use values for fsetpos() that are returned from fgetpos()	No
CERT-C:FIO45-C	Avoid TOCTOU race conditions while accessing files	Yes
CERT-C:FIO46-C	Do not access a closed file	Yes
CERT-C:FIO47-C	Use valid format strings	Yes
CERT-C:FLP00-C	Understand the limitations of floating-point numbers	No
CERT-C:FLP01-C	Take care in rearranging floating-point expressions	No
CERT-C:FLP02-C	Avoid using floating-point numbers when precise computation is needed	No
CERT-C:FLP03-C	Detect and handle floating-point errors	No
CERT-C:FLP04-C	Check floating-point inputs for exceptional values	No
CERT-C:FLP05-C	Do not use denormalized numbers	No
CERT-C:FLP06-C	Convert integers to floating point for floating-point operations	Yes
CERT-C:FLP07-C	Cast the return value of a function that returns a floating-point type	No
CERT-C:FLP30-C	Do not use floating-point variables as loop counters	Yes
CERT-C:FLP32-C	Prevent or detect domain and range errors in math functions	Yes
CERT-C:FLP34-C	Ensure that floating-point conversions are within range of the new type	Yes
CERT-C:FLP36-C	Preserve precision when converting integral values to floating-point type	Yes
CERT-C:FLP37-C	Do not use object representations to compare floating-point values	No
CERT-C:INT00-C	Understand the data model used by your implementation(s)	No
CERT-C:INT01-C	Use rsize_t or size_t for all integer values representing the size of an object	No
CERT-C:INT02-C	Understand integer conversion rules	Yes
CERT-C:INT04-C	Enforce limits on integer values originating from tainted sources	Yes
CERT-C:INT05-C	Do not use input functions to convert character data if they cannot handle all possible inputs	Yes
CERT-C:INT07-C	Use only explicitly signed or unsigned char type for numeric values	Yes
CERT-C:INT08-C	Verify that all integer values are in range	Yes
CERT-C:INT09-C	Ensure enumeration constants map to unique values	Yes
CERT-C:INT10-C	Do not assume a positive remainder when using the % operator	No
CERT-C:INT12-C	Do not make assumptions about the type of a plain int bit-field when used in an expression	Yes
CERT-C:INT13-C	Use bitwise operators only on unsigned operands	Yes
CERT-C:INT14-C	Avoid performing bitwise and arithmetic operations on the same data	No
CERT-C:INT15-C	Use intmax_t or uintmax_t for formatted IO on programmer-defined integer types	No
CERT-C:INT16-C	Do not make assumptions about representation of signed integers	No

CERT-C:INT17-C	Define integer constants in an implementation-independent manner	No
CERT-C:INT18-C	Evaluate integer expressions in a larger size before comparing or assigning to that size	Yes
CERT-C:INT30-C	Ensure that unsigned integer operations do not wrap	Yes
CERT-C:INT31-C	Ensure that integer conversions do not result in lost or misinterpreted data	Yes
CERT-C:INT32-C	Ensure that operations on signed integers do not result in overflow	Yes
CERT-C:INT33-C	Ensure that division and remainder operations do not result in divide-by-zero errors	Yes
CERT-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	Yes
CERT-C:INT35-C	Use correct integer precisions	Yes
CERT-C:INT36-C	Converting a pointer to integer or integer to pointer	Yes
CERT-C:MEM00-C	Allocate and free memory in the same module, at the same level of abstraction	Yes
CERT-C:MEM01-C	Store a new value in pointers immediately after free()	Yes
CERT-C:MEM02-C	Immediately cast the result of a memory allocation function call into a pointer to the allocated type	No
CERT-C:MEM03-C	Clear sensitive information stored in reusable resources	No
CERT-C:MEM04-C	Beware of zero-length allocations	No
CERT-C:MEM05-C	Avoid large stack allocations	Yes
CERT-C:MEM06-C	Ensure that sensitive data is not written out to disk	No
CERT-C:MEM07-C	Ensure that the arguments to calloc(), when multiplied, do not wrap	Yes
CERT-C:MEM10-C	Define and use a pointer validation function	No
CERT-C:MEM11-C	Do not assume infinite heap space	Yes
CERT-C:MEM12-C	Consider using a goto chain when leaving a function on error when using and releasing resources	No
CERT-C:MEM30-C	Do not access freed memory	Yes
CERT-C:MEM31-C	Free dynamically allocated memory when no longer needed	Yes
CERT-C:MEM33-C	Allocate and copy structures containing a flexible array member dynamically	Yes
CERT-C:MEM34-C	Only free memory allocated dynamically	Yes
CERT-C:MEM35-C	Allocate sufficient memory for an object	Yes
CERT-C:MEM36-C	Do not modify the alignment of objects by calling realloc()	Yes
CERT-C:MSC00-C	Compile cleanly at high warning levels	Yes
CERT-C:MSC01-C	Strive for logical completeness	No
CERT-C:MSC04-C	Use comments consistently and in a readable fashion	No
CERT-C:MSC05-C	Do not manipulate time_t typed values directly	No
CERT-C:MSC06-C	Beware of compiler optimizations	Yes
CERT-C:MSC07-C	Detect and remove dead code	Yes
CERT-C:MSC09-C	Character encoding: Use subset of ASCII for safety	No
CERT-C:MSC10-C	Character encoding: UTF8-related issues	No
CERT-C:MSC11-C	Incorporate diagnostic tests using assertions	Yes
CERT-C:MSC12-C	Detect and remove code that has no effect or is never executed	Yes
CERT-C:MSC13-C	Detect and remove unused values	Yes
CERT-C:MSC14-C	Do not introduce unnecessary platform dependencies	No
CERT-C:MSC15-C	Do not depend on undefined behavior	No
CERT-C:MSC17-C	Finish every set of statements associated with a case label with a break statement	Yes
CERT-C:MSC18-C	Be careful while handling sensitive data, such as passwords, in program code	Yes
CERT-C:MSC19-C	For functions that return an array, prefer returning an empty array over a null value	No
CERT-C:MSC20-C	Do not use a switch statement to transfer control into a complex block	Yes
CERT-C:MSC21-C	Use robust loop termination conditions	Yes
CERT-C:MSC22-C	Use the setjmp(), longjmp() facility securely	Yes
CERT-C:MSC23-C	Beware of vendor-specific library and language differences	No
CERT-C:MSC24-C	Do not use deprecated or obsolescent functions	Yes

CERT-C:MSC30-C	Do not use the rand() function for generating pseudorandom numbers	Yes
CERT-C:MSC32-C	Properly seed pseudorandom number generators	Yes
CERT-C:MSC33-C	Do not pass invalid data to the asctime() function	Yes
CERT-C:MSC37-C	Ensure that control never reaches the end of a non-void function	Yes
CERT-C:MSC38-C	Do not treat a predefined identifier as an object if it might only be implemented as a macro	Yes
CERT-C:MSC39-C	Do not call va_arg() on a va_list that has an indeterminate value	Yes
CERT-C:MSC40-C	Do not violate constraints	No
CERT-C:MSC41-C	Never hard code sensitive information	Yes
CERT-C:POS01-C	Check for the existence of links when dealing with files	No
CERT-C:POS02-C	Follow the principle of least privilege	No
CERT-C:POS04-C	Avoid using PTHREAD_MUTEX_NORMAL type mutex locks	No
CERT-C:POS05-C	Limit access to files by creating a jail	Yes
CERT-C:POS30-C	Use the readlink() function properly	Yes
CERT-C:POS33-C	Do not use vfork()	Yes
CERT-C:POS34-C	Do not call putenv() with a pointer to an automatic variable as the argument	Yes
CERT-C:POS35-C	Avoid race conditions while checking for the existence of a symbolic link	No
CERT-C:POS36-C	Observe correct revocation order while relinquishing privileges	No
CERT-C:POS37-C	Ensure that privilege relinquishment is successful	No
CERT-C:POS38-C	Beware of race conditions when using fork and file descriptors	Yes
CERT-C:POS39-C	Use the correct byte ordering when transferring data between systems	No
CERT-C:POS44-C	Do not use signals to terminate threads	Yes
CERT-C:POS47-C	Do not use threads that can be canceled asynchronously	No
CERT-C:POS48-C	Do not unlock or destroy another POSIX thread's mutex	Yes
CERT-C:POS49-C	When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed	Yes
CERT-C:POS50-C	Declare objects shared between POSIX threads with appropriate storage durations	No
CERT-C:POS51-C	Avoid deadlock with POSIX threads by locking in predefined order	Yes
CERT-C:POS52-C	Do not perform operations that can block while holding a POSIX lock	Yes
CERT-C:POS53-C	Do not use more than one mutex for concurrent waiting operations on a condition variable	No
CERT-C:POS54-C	Detect and handle POSIX library errors	Yes
CERT-C:PRE00-C	Prefer inline or static functions to function-like macros	Yes
CERT-C:PRE01-C	Use parentheses within macros around parameter names	No
CERT-C:PRE02-C	Macro replacement lists should be parenthesized	Yes
CERT-C:PRE03-C	Prefer typedefs to defines for encoding non-pointer types	No
CERT-C:PRE04-C	Do not reuse a standard header file name	No
CERT-C:PRE05-C	Understand macro replacement when concatenating tokens or performing stringification	Yes
CERT-C:PRE06-C	Enclose header files in an include guard	No
CERT-C:PRE07-C	Avoid using repeated question marks	No
CERT-C:PRE08-C	Guarantee that header file names are unique	No
CERT-C:PRE09-C	Do not replace secure functions with deprecated or obsolescent functions	No
CERT-C:PRE10-C	Wrap multistatement macros in a do-while loop	No
CERT-C:PRE11-C	Do not conclude macro definitions with a semicolon	Yes
CERT-C:PRE12-C	Do not define unsafe macros	No
CERT-C:PRE13-C	Use the Standard predefined macros to test for versions and features.	No
CERT-C:PRE30-C	Do not create a universal character name through concatenation	Yes
CERT-C:PRE31-C	Avoid side effects in arguments to unsafe macros	Yes
CERT-C:PRE32-C	Do not use preprocessor directives in invocations of function-like macros	Yes
CERT-C:SIG00-C	Mask signals handled by noninterruptible signal handlers	Yes
CERT-C:SIG01-C	Understand implementation-specific details regarding signal handler persistence	Yes

CERT-C:SIG02-C	Avoid using signals to implement normal functionality	Yes
CERT-C:SIG30-C	Call only asynchronous-safe functions within signal handlers	Yes
CERT-C:SIG31-C	Do not access shared objects in signal handlers	Yes
CERT-C:SIG34-C	Do not call signal() from within interruptible signal handlers	Yes
CERT-C:SIG35-C	Do not return from a computational exception signal handler	Yes
CERT-C:STR00-C	Represent characters using an appropriate type	Yes
CERT-C:STR01-C	Adopt and implement a consistent plan for managing strings	No
CERT-C:STR02-C	Sanitize data passed to complex subsystems	Yes
CERT-C:STR03-C	Do not inadvertently truncate a string	Yes
CERT-C:STR04-C	Use plain char for characters in the basic character set	Yes
CERT-C:STR05-C	Use pointers to const when referring to string literals	Yes
CERT-C:STR06-C	Do not assume that strtok() leaves the parse string unchanged	No
CERT-C:STR07-C	Use the bounds-checking interfaces for string manipulation	Yes
CERT-C:STR08-C	Use managed strings for development of new string manipulation code	No
CERT-C:STR09-C	Don't assume numeric values for expressions with type plain character	No
CERT-C:STR10-C	Do not concatenate different type of string literals	No
CERT-C:STR11-C	Do not specify the bound of a character array initialized with a string literal	No
CERT-C:STR30-C	Do not attempt to modify string literals	Yes
CERT-C:STR31-C	Guarantee that storage for strings has sufficient space for character data and the null terminator	Yes
CERT-C:STR32-C	Do not pass a non-null-terminated character sequence to a library function that expects a string	Yes
CERT-C:STR34-C	Cast characters to unsigned char before converting to larger integer sizes	Yes
CERT-C:STR37-C	Arguments to character-handling functions must be representable as an unsigned char	Yes
CERT-C:STR38-C	Do not confuse narrow and wide character strings and functions	Yes
CERT-C:WIN00-C	Be specific when dynamically loading libraries	Yes
CERT-C:WIN01-C	Do not forcibly terminate execution	No
CERT-C:WIN02-C	Restrict privileges when spawning child processes	Yes
CERT-C:WIN03-C	Understand HANDLE inheritance	No
CERT-C:WIN04-C	Consider encrypting function pointers	No
CERT-C:WIN30-C	Properly pair allocation and deallocation functions	Yes