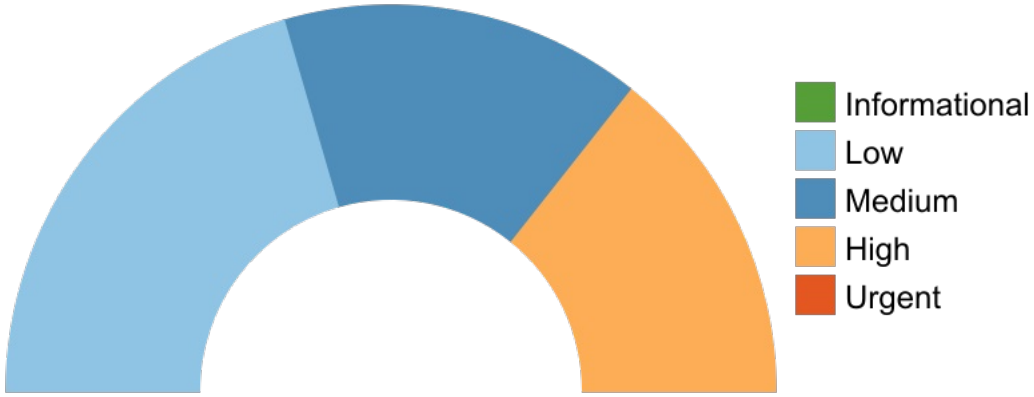
This standard provides rules for secure coding in the C programming language.

The rules and recommendations in this standard are a work in progress and reflect the current thinking of the secure coding community. As rules and recommendations mature, they are published in report or book form as official releases. These releases are issued as dictated by the needs and interests of the secure software development community.

The list of rules and recommendations in this tool were last updated on 2023/05/23.

Checks by Severity



Checks

Check ID	Check Name	Supported	Severity
ARR30-C	Do not form or use out-of-bounds pointers or array subscripts	No	High
ARR32-C	Ensure size arguments for variable length arrays are in a valid range	No	High
ARR36-C	Do not subtract or compare two pointers that do not refer to the same array	Yes	Medium
ARR37-C	Do not add or subtract an integer to a pointer to a non-array object	Yes	Medium
ARR38-C	Guarantee that library functions do not form invalid pointers	No	High
ARR39-C	Do not add or subtract a scaled integer to a pointer	Yes	High
CON30-C	Clean up thread-specific storage	Yes	Medium
CON31-C	Do not destroy a mutex while it is locked	Yes	Medium
CON32-C	Prevent data races when accessing bit-fields from multiple threads	No	Medium

CON33-C	Avoid race conditions when using library functions	Yes	Medium
CON34-C	Declare objects shared between threads with appropriate storage durations	No	Medium
CON35-C	Avoid deadlock by locking in a predefined order	No	Low
CON36-C	Wrap functions that can spuriously wake up in a loop	Yes	Low
CON37-C	Do not call signal() in a multithreaded program	Yes	Low
CON38-C	Preserve thread safety and liveness when using condition variables	Yes	Low
CON39-C	Do not join or detach a thread that was previously joined or detached	Yes	Low
CON40-C	Do not refer to an atomic variable twice in an expression	Yes	Medium
CON41-C	Wrap functions that can fail spuriously in a loop	Yes	Low
CON43-C	Do not allow data races in multithreaded code	No	Medium
DCL30-C-A	Declare objects with appropriate storage durations - assigning addresses	Yes	High
DCL30-C-B	Declare objects with appropriate storage durations - returning addresses	Yes	High
DCL31-C	Declare identifiers before using them	Yes	Low
DCL36-C	Do not declare an identifier with conflicting linkage classifications	Yes	Medium
DCL37-C	Do not declare or define a reserved identifier	Yes	Low
DCL38-C	Use the correct syntax when declaring a flexible array member	Yes	Low
DCL39-C	Avoid information leakage when passing a structure across a trust boundary	No	Low
DCL40-C	Do not create incompatible declarations of the same function or object	Yes	Low
DCL41-C	Do not declare variables inside a switch statement before the first case label	Yes	Medium
ENV30-C	Do not modify the object referenced by the return value of certain functions	Yes	Low
ENV31-C	Do not rely on an environment pointer following an operation that may invalidate it	Yes	Low

ENV32-C	All exit handlers must return normally	Yes	Medium
ENV33-C	Do not call system()	Yes	High
ENV34-C	Do not store pointers returned by certain functions	Yes	Low
ERR30-C	Take care when reading errno	Yes	Medium
ERR32-C	Do not rely on indeterminate values of errno	No	Low
ERR33-C	Detect and handle standard library errors	Yes	High
ERR34-C	Detect errors when converting a string to a number	Yes	Medium
EXP30-C-A	Do not depend on the order of evaluation for side effects - calls	Yes	Medium
EXP30-C-B	Do not depend on the order of evaluation for side effects - other	Yes	Medium
EXP32-C	Do not access a volatile object through a nonvolatile reference	Yes	Low
EXP33-C	Do not read uninitialized memory	Yes	High
EXP34-C	Do not dereference null pointers	Yes	High
EXP35-C	Do not modify objects with temporary lifetime	No	Low
EXP36-C	Do not cast pointers into more strictly aligned pointer types	Yes	Low
EXP37-C	Call functions with the correct number and type of arguments	Yes	Medium
EXP39-C	Do not access a variable through a pointer of an incompatible type	Yes	Medium
EXP40-C	Do not modify constant objects	No	Low
EXP42-C	Do not compare padding data	Yes	Medium
EXP43-C	Avoid undefined behavior when using restrict-qualified pointers	No	Medium
EXP44-C	Do not rely on side effects in operands to sizeof, _Alignof, or _Generic	Yes	Low
EXP45-C	Do not perform assignments in selection statements	Yes	Low
EXP46-C	Do not use a bitwise operator with a Boolean-like operand	Yes	Low
EXP47-C	Do not call va_arg with an argument of the incorrect type	Yes	Medium
FIO30-C	Exclude user input from format strings	Yes	High
FIO32-C	Do not perform operations on devices that are only appropriate for files	No	Medium
FIO34-C	Distinguish between characters read from a	No	High

	file and EOF or WEOF		
FIO37-C	Do not assume that fgets() or fgetws() returns a nonempty string when successful	Yes	High
FIO38-C	Do not copy a FILE object	Yes	Low
FIO39-C	Do not alternately input and output from a stream without an intervening flush or positioning call	Yes	Low
FIO40-C	Reset strings on fgets() or fgetws() failure	Yes	Low
FIO41-C	Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects	Yes	Low
FIO42-C	Close files when they are no longer needed	Yes	Medium
FIO44-C	Only use values for fsetpos() that are returned from fgetpos()	Yes	Medium
FIO45-C	Avoid TOCTOU race conditions while accessing files	Yes	High
FIO46-C	Do not access a closed file	Yes	Medium
FIO47-C	Use valid format strings	Yes	High
FLP30-C	Do not use floating-point variables as loop counters	Yes	Low
FLP32-C	Prevent or detect domain and range errors in math functions	No	Medium
FLP34-C	Ensure that floating-point conversions are within range of the new type	No	Low
FLP36-C	Preserve precision when converting integral values to floating-point type	No	Low
FLP37-C	Do not use object representations to compare floating-point values	Yes	Low
INT30-C	Ensure that unsigned integer operations do not wrap	Yes	High
INT31-C	Ensure that unsigned integer operations do not result in lost or misinterpreted data	Yes	High
INT32-C	Ensure that operations on signed integers do not result in overflow	No	High
INT33-C	Division by Zero	Yes	Low
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	No	Low
INT35-C	Use correct integer precisions	No	Low
INT36-C	Converting a pointer to integer or integer to	Yes	Low

	pointer		
MEM30-C	Do not access freed memory	No	High
MEM31-C	Free dynamically allocated memory when no longer needed	Yes	Medium
MEM33-C	Allocate and copy structures containing a flexible array member dynamically	Yes	Low
MEM34-C	Only free memory allocated dynamically	Yes	High
MEM35-C	Allocate sufficient memory for an object	Yes	High
MEM36-C	Do not modify the alignment of objects by calling realloc()	No	Low
MSC30-C	Do not use the rand() function for generating pseudorandom numbers	Yes	Medium
MSC32-C	Properly seed pseudorandom number generators	Yes	Medium
MSC33-C	Do not pass invalid data to the asctime() function	Yes	High
MSC37-C	Ensure that control never reaches the end of a non-void function	Yes	High
MSC38-C	Do not treat a predefined identifier as an object if it might only be implemented as a macro	Yes	Low
MSC39-C	Do not call va_arg() on a va_list that has an indeterminate value	Yes	Low
MSC40-C	Do not violate constraints	Yes	Low
MSC41-C	Never hard code sensitive information	No	High
POS30-C	Use the readlink() function properly	Yes	High
POS34-C	Do not call putenv() with a pointer to an automatic variable as the argument	Yes	High
POS35-C	Avoid race conditions while checking for the existence of a symbolic link	Yes	High
POS36-C	Observe correct revocation order while relinquishing privileges	Yes	High
POS37-C	Ensure that privilege relinquishment is successful	Yes	High
POS38-C	Beware of race conditions when using fork and file descriptors	Yes	Medium
POS39-C	Use the correct byte ordering when transferring data between systems	Yes	Medium
POS44-C	Do not use signals to terminate threads	Yes	Low
POS47-C	Do not use threads that can be canceled asynchronously	Yes	Medium

POS48-C	Do not unlock or destroy another POSIX thread's mutex	Yes	Medium
POS49-C	When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed	No	Medium
POS50-C	Declare objects shared between POSIX threads with appropriate storage durations	Yes	Medium
POS51-C	Avoid deadlock with POSIX threads by locking in predefined order	Yes	Low
POS52-C	Do not perform operations that can block while holding a POSIX lock	Yes	Low
POS53-C	Do not use more than one mutex for concurrent waiting operations on a condition variable	Yes	Medium
POS54-C	Detect and handle POSIX library errors	Yes	High
PRE30-C	Do not create a universal character name through concatenation	Yes	Low
PRE31-C	Avoid side effects in arguments to unsafe macros	Yes	Low
PRE32-C	Do not use preprocessor directives in invocations of function-like macros	Yes	Low
SIG30-C	Call only asynchronous-safe functions within signal handlers	Yes	High
SIG31-C	Do not access shared objects in signal handlers	Yes	High
SIG34-C	Do not call signal() from within interruptible signal handlers	Yes	Low
SIG35-C	Do not return from a computational exception signal handler	No	Low
STR30-C	Do not attempt to modify string literals	Yes	Low
STR31-C	Guarantee that storage for strings has sufficient space for character data and the null terminator	Yes	High
STR32-C	Null-terminated strings passed to library functions	Yes	High
STR34-C	Cast characters to unsigned char before converting to larger integer sizes	No	Medium
STR37-C	Arguments to character-handling functions must be representable as an unsigned char	Yes	Low
STR38-C	Do not confuse narrow and wide character strings and functions	Yes	High

WIN30-C	Properly pair allocation and deallocation functions	Yes	Low