HKMC Secure C

Perforce QAC for C 2026.1

(HKMC Secure C Rev. Date '21.04 Release 4.1)

Rule Enforcement Summary

		Total
a	Total Number of Rules	129
b	Total Number of 'Not Statically Enforceable' Rules (Assisted/Unassisted)	0
c	Total Number of Enforceable Rules (a-b)	129
d	Total Number of Enforced Rules	125
e	Total Number of Unenforced Rules (c-d)	4
f	Enforced Rules Percentage (d/c)	97%
g	Unenforced Rules Percentage (e/c)	3%

Rule Enforcement

Id	Description	Severity	Enforced
Code Error			
3.1 Preprocessor			
C-PRE-001	Do not create a character name of Universal character set (UCS) through token concatenation	Low	Yes
C-PRE-002	Do not pass specific variable value conversion argument to unsafe macros	Low	Yes
C-PRE-003	Do not pass preprocessing directives in when calling function-like macros	Low	Yes
3.2 Declaration and Initialization			
C-DCI-001	Do not returning and assigning local variable address	High	Yes
C-DCI-002	Do not declare same variable for internal and external linkage	Medium	Yes

Id	Description	Severity	Enforced
C-DCI-003	Do not declare or define a reserved identifier	Low	Yes
C-DCI-004	Need to use the correct syntax when declaring a flexible array	Low	Yes
C-DCI-005	Need to avoid information leakage when passing a structure across a trust boundary	Low	Yes
C-DCI-006	Do not declare of incompatible function or object	Low	Yes
C-DCI-007	Do not declare variable in a switch statement before the first case label	Medium	Yes
C-DCI-008	Attention to switch race condition	Medium	Yes
C-DCI-009	Do not miss default case in switch statement	Medium	Yes
C-DCI-010	Do not omit break in switch statement	Medium	Yes
C-DCI-011	Need explicit handling of operator precedence using parentheses [MCU]	High	Yes
C-DCI-012	Do not compare numerical ranges without minimum check [MCU]	High	Yes
3.3 Integers, Characters and Strings			
C-FLP-001	Do not use floating-point variables as loop counters	Low	Yes
C-FLP-002	Ensure that floating-point type conversion occurs within the range of new types	Low	Yes
C-FLP-003	Preserve precision when converting from integer type to floating-point type	Low	Yes
C-INT-001	Attention to integer conversion rules (CERT Recommendations)	Medium	Yes
C-INT-002	Ensure that unsigned integer operations do not wrap	High	Yes
C-INT-003	Integer conversion must be guaranteed not to result in lost or misinterpreted data [AP, MCU]	High	Yes
C-INT-004	Ensure that signed integer operations do not overflow [AP, MCU]	High	Yes
C-INT-005	Ensure that division and modulo operations do not result in 'divide-by-zero' errors [AP, MCU]	Low	Yes
C-INT-006	Do not shift to negative values or shift by more than or equal to the bits of the operands	Low	Yes
C-INT-007	Use the correct precisions of integer type	Low	Yes

Id	Description	Severity	Enforced
C-INT-008	Attention to type conversion between pointer and integer [AP, MCU]	Low	Yes
C-INT-009	Attention to calculate for buffer or data type size	High	Yes
C-STR-001	Do not modify string literals	Low	Yes
C-STR-002	Ensure that storage has sufficiently space for character data and the null terminator	High	Yes
C-STR-003	Do not passing a non-null-terminated string to a library function	High	Yes
C-STR-004	Argument to character-handling function must be passed as unsigned char	Low	Yes
C-STR-005	Do not confuse narrow and wide character strings when passing arguments to character-handling function	High	Yes
C-STR-006	Incorrect calculation of the multi-byte string length	Low	Yes
3.4 Expressions			
C-EXP-001	Do not ignore return value by functions (CERT Recommendations) [MCU]	Medium	Yes
C-EXP-002	Attention to the order of calculation (evaluation) that causes the side effects	Medium	Yes
C-EXP-003	Do not reference(access, read) uninitialized memory [Ap, MCU]	High	Yes
C-EXP-004	Attention not to reference abnormal pointer [MCU]	Medium	Yes
C-EXP-005	Attention when using assignment operator instead of comparison operators [MCU]	Low	Yes
C-EXP-006	Attention when using comparison operators instead assignment operators [MCU]	Medium	Yes
C-EXP-007	Attention to external input not checked for the control condition of the loop [MCU]	High	Yes
C-EXP-008	Do not use possible recursion calls and only allow a limited number of recursion calls if necessary	Medium	Yes
C-EXP-009	Do not null pointer dereference [AP, MCU]	High	Yes
C-EXP-010	Do not cast pointer into more strictly aligned pointer types	Low	Yes
C-EXP-011	Do not variable access through pointers of incompatible types	Medium	Yes

Id	Description	Severity	Enforced
C-EXP-012	Do not modify constant object	Low	Yes
C-EXP-013	Do not compare padding data	Medium	Yes
C-EXP-014	Attention to avoid undefined behavior when using a pointer that qualifies restrict	Medium	Yes
C-EXP-015	Do not pass an expression that changes the value of variable as an operand to sizeof, _Alignof, _Generic	Low	Yes
C-EXP-016	Do not perform assignment operation on certain operators	Low	Yes
C-EXP-017	Do not use a bitwise operator with Boolean operand	Low	Yes
C-EXP-018	Do not call va_arg with an argument of the incorrect type	Medium	Yes
C-EXP-019	Do not use of Path Manipulation Function without Maximum-sized Buffer	Medium	Yes
3.5 Arrays			
C-ARR-001	Ensure that the index is within the valid range for memory read and write operations of the array [AP, MCU]	High	Yes
C-ARR-002	Ensure size arguments for variable length array are in a valid range	High	Yes
C-ARR-003	Do not subtract or compare two pointers that do not reference the same array [AP, MCU]	Medium	Yes
C-ARR-004	Do not add or subtract an integer to a pointer to a non-array object [Ap, MCU]	Medium	Yes
C-ARR-005	Do not generate invalid pointer through library functions	High	Yes
C-ARR-006	Do not add and subtract integers resized automatically in pointer [AP, MCU]	High	Yes
C-ARR-007	Attention for handling length parameters when accessing arrays in loops like the for statements [MCU]	Medium	Yes
C-ARR-008	Write before buffer start	High	Yes
C-ARR-009	Buffer access using source buffer size	Low	Yes
C-ARR-010	Attention not to include invalid arguments when calling a function [MCU]	Medium	Yes
C-ARR-011	Attention for incorrect judgement of return value by function [MCU]	Medium	Yes

Id	Description	Severity	Enforced
3.6 Error handling			
C-ERR-001	Attention when calling a library related the errno	Medium	Yes
C-ERR-002	Detect and handle errors of standard library	High	Yes
C-ERR-003	Detect errors when converting a string to a number	Medium	Yes
Code quality			
4.1 Memory Management			
C-MEM-001	Do not reference freed memory	High	Yes
C-MEM-002	Free dynamically allocated memory when no longer needed	Medium	Yes
C-MEM-003	Allocate and copy structures containing a flexible array member dynamically	Low	Yes
C-MEM-004	Free memory allocated dynamically	High	Yes
C-MEM-005	Allocate sufficient memory for an object	High	Yes
C-MEM-006	Do not modify the alignment of object by calling realloc()	Low	Yes
C-MEM-007	Improper clearing of heap memory before release('Heap inspection')	Medium	Yes
4.2 Input and output			
C-FIO-001	Exclude user input from format strings	High	Yes
C-FIO-002	Distinguish between characters read from a file and EOF or WEOF	High	Yes
C-FIO-003	Do not assume that fgets() or fgetws() returns a nonempty string when successful	High	Yes
C-FIO-004	Do not input and output from a stream without an intervening flush or positioning call	Low	Yes
C-FIO-005	Reset strings on fgets() or fgetws() failure	Low	Yes
C-FIO-006	Do not call getc(), putc(), getwc() or putwc() with a string argument that changes the value of a variable	Low	Yes
C-FIO-007	Close file when they are no longer needed	Medium	Yes
C-FIO-008	Use valid format strings	High	Yes

Id	Description	Severity	Enforced
C-FIO-009	Do not concurrent execution use shared resource with improper synchronization(Race condition)	Low	Yes
4.3 Environment variable			
C-ENV-001	Do not modify the object referenced by the return value of certain functions	Low	Yes
C-ENV-002	Do not reference the environment variable pointer after a function is called that contains an operation that invalidates the environment variable pointer	Low	Yes
C-ENV-003	All exit handlers must return normally	Medium	Yes
C-ENV-004	Do not call system()	High	Yes
4.4 Signals			
C-SIG-001	Only call functions that are asynchronously safe in the signal handler	High	Yes
C-SIG-002	Do not return from a computational exception signal handler	Low	Yes
4.5 Concurrency programming			
C-CON-001	Clean up thread storage after allocation	Medium	Yes
C-CON-002	Do not destroy a mutex while it is locked	Medium	Yes
C-CON-003	Prevent data race when accessing bit-fields from multiple threads	Medium	Yes
C-CON-004	Avoid race conditions when using library functions	Medium	Yes
C-CON-005	Declare object shared between threads	Medium	Yes
C-CON-006	Lock according to predefine order to avoid deadlock	Low	Yes
C-CON-007	Wrapping of functions that can spuriously wake up in a loop	Low	Yes
C-CON-008	Do not call signal() in a multithreaded program	Low	Yes
C-CON-009	Preserve thread safety and liveness when using condition variables	Low	Yes
C-CON-010	Do not join or detach a thread that was previously joined or detached	Low	Yes
C-CON-011	Do not reference to an atomic variable twice in an expression	Medium	Yes
Security functions			

Id	Description	Severity	Enforced
5.1 Security functions			
C-MSC-001	Should be treated as 2^16 + 1, ie 65537 in the public key index when using asymmetric key RSA	High	Yes
C-MSC-002	Use at least 8-Byte when using MAC	High	Yes
C-MSC-003	Do not use the rand() when generating random number using PRNG	Medium	Yes
C-MSC-004	Set seeds correctly when using PRNG	Medium	Yes
C-MSC-005	Attention to pass improper argument when using the asctime()	High	Yes
C-MSC-006	Return all functions other than void return type	High	Yes
C-MSC-007	Do not access to identifier as an object when a predefined identifier is defined only in macro	Low	Yes
C-MSC-008	Do not call va_arg() on va_list with an unspecified value	Low	Yes
C-MSC-009	Code removal error clearing compiler's buffer	Medium	Yes
C-MSC-010	Do not use getloin in multithreaded applications	Medium	Yes
C-MSC-011	Do not transmit and store critical security and vehicle information to outside controller in the form of plain text [MCU]	High	Yes
C-MSC-012	Replace periodically the secret key and keep confidential of updated key[MCU]	High	Yes
C-MSC-013	Satisfy a certain level of security when using cryptographic algorithms or hash functions	High	No
C-MSC-014	Do not use algorithms that find security vulnerabilities, such as the DES symmetric key algorithm and the MD5 hash algorithm [MCU]	High	Yes
C-MSC-015	Do not use fixed or predictable seeds [MCU]	High	Yes
C-MSC-016	Include a message identification value to prevent replay attacks when creating a MAC to ensure integrity	High	No
C-MSC-017	Validate the certificate and signature of the public key when the sender sends a message using asymmetric key algorithm [MCU]	High	No
C-MSC-018	Store critical security information, such as key values, in a secure memory space [MCU]	High	No
5.2 POSIX			

Id	Description	Severity	Enforced
C-POS-001	Correct use of readlink()	High	Yes
C-POS-002	Do not use for vfork()	Low	Yes
C-POS-003	Do not pass a pointer to a local variable in the putenv()	High	Yes
C-POS-004	Attention for racing conditions when using fork and file descriptors	Medium	Yes
C-POS-005	Use of correct byte order is required for data communication between systems	Medium	Yes
C-POS-006	Do not use signals to terminate threads	Low	Yes
C-POS-007	Do not unlock another POSIX thread's mutex	Medium	Yes
C-POS-008	Detect and handle POSIX Library errors	High	Yes
C-POS-009	Attention improper resource locking	High	Yes
C-POS-010	Attention to leave without development mode, including debug code [MCU]	High	Yes
C-POS-011	Clear block delimitation of the statements [MCU]	Medium	Yes
C-POS-012	Remove 'Dead Code' [MCU]	Medium	Yes