Flawnter SAST MISRA C2012 Rule Coverage

MISRA C is a set of software development guidelines developed by The MISRA Consortium

Static Application Security Testing (SAST) is a crucial process in modern software development, designed to identify quality and security flaws within the source code of applications before they are deployed. Flawnter SAST MISRA rule sets are predefined sets of rules that Flawnter use to analyze and evaluate source code for issues that violate MISRA guidelines. These rules play an important role in the SAST process, as they define the criteria for identifying quality issues, logical errors, reliability, readability and other issues that may cause the application to crash or behave abnormally. In the table below we have listed the rules supported by Flawnter.

Rule Name	Description	Category	C2012
Directive 4.3	Assembly language shall be encapsulated and isolated.	Required	Yes
Directive 4.4	Sections of code should not be "commented out".	Advisory	Yes
Directive 4.5	Identifiers in the same name space with overlapping visibility should be typographically unambiguous.	Advisory	Yes
Directive 4.6	Typedefs that indicate size and signedness should be used in place of the basic numerical types.	Advisory	Yes
Directive 4.7	If a function returns error information, then that error information shall be tested.	Required	Yes
Directive 4.9	A function should be used in preference to a function-like macro where they are interchangeable.	Advisory	Yes
Directive 4.10	Precautions shall be taken in order to prevent the contents of a header file being included more than once.	Required	Yes
Directive 4.12	Dynamic memory allocation shall not be used.	Required	Yes
Directive 4.13	Functions which are designed to provide operations on a resource should be called in an appropriate sequence.	Advisory	Yes
Directive 4.14	The validity of values received from external sources shall be checked.	Required	No
Directive 4.15	Evaluation of floating-point expressions shall not lead to the undetected generation of infinities and NaNs.	Required	No
Directive 5.1	There shall be no data races between threads.	Required	No
Directive 5.2	There shall be no deadlocks between threads.	Required	No
Rule 1.1	The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits.	Required	Yes
Rule 1.2	Language extensions should not be used.	Advisory	Yes
Rule 1.4	Emergent language features shall not be used.	Required	No
Rule 1.5	Obsolescent language features shall not be used.	Required	No
Rule 2.1	A project shall not contain unreachable code.	Required	Yes
Rule 2.2	There shall be no dead code.	Required	Yes
Rule 2.4	A project should not contain unused ta g declarations.	Advisory	Yes

Rule 2.6	A function should not contain unused label declarations.	Advisory	Yes
Rule 3.2	Line-splicing shall not be used in // comments.	Required	Yes
Rule 4.1	Octal and hexadecimal escape sequences shall be terminated.	Required	Yes
Rule 4.2	Trigraphs should not be used.	Advisory	Yes
Rule 5.1	External identifiers shall be distinct.	Required	Yes
Rule 5.2	Identifier s declared in the same scope and name space shall be distinct.	Required	Yes
Rule 5.4	Macro identifier s shall be distinct.	Required	Yes
Rule 5.6	A typedef name shall be a unique identifier.	Required	Yes
Rule 6.1	Bit-fields shall only be declared with an appropriate type.	Required	Yes
Rule 6.2	Single-bit named bit fields shall not be of a signed type.	Required	Yes
Rule 7.1	Octal constants shall not be used.	Required	Yes
Rule 7.3	The lowercase character “l” shall not be used in a literal suffix.	Required	Yes
Rule 7.4	A string literal shall not be assigned to an object unless the object’s type is “pointer to const-qualified char”.	Required	Yes
Rule 8.1	Types shall be explicitly specified.	Required	Yes
Rule 8.2	Function types shall be in prototype form with named parameters.	Required	Yes
Rule 8.8	The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage.	Required	Yes
Rule 8.10	An inline function shall be declared with the static storage class.	Required	Yes
Rule 8.11	When an array with external linkage is declared, its size should be explicitly specified.	Advisory	Yes
Rule 8.12	Within a n enumerator list, the value of an implicitly-specified enumeration constant shall be unique.	Required	Yes
Rule 8.14	The restrict type qualifier shall not be used.	Required	Yes
Rule 9.3	Arrays shall not be partially initialized.	Required	Yes
Rule 9.5	Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly.	Required	Yes
Rule 10.2	Expressions with essentially character type (character data) shall not be used arithmetically as the data does not represent numeric values.	Required	Yes
Rule 11.9	The macro NULL shall be the only permitted form of integer null pointer constant.	Required	Yes
Rule 12.1	The precedence of operators within expression s should be made explicit.	Advisory	Yes
Rule 12.3	The comma operator should not be used.	Advisory	Yes
Rule 13.2	The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders	Required	Yes
Rule 13.3	A full expression containing an increment (++) or decrement (--) operator should have no other potential side effects other than that caused by the increment or decrement operator.	Advisory	Yes
Rule 13.4	The result of an assignment operator should not be used.	Advisory	Yes

Rule 13.6	The operand of the sizeof operator shall not contain any expression which has potential side effects.	Mandatory	Yes
Rule 14.1	A loop counter shall not have essentially floating type.	Required	Yes
Rule 14.4	The controlling expression of an if statement and the controlling expression of an iteration- statement shall have essentially Boolean type.	Required	Yes
Rule 15.1	The goto statement should not be used.	Advisory	Yes
Rule 15.2	The goto statement shall jump to a label declared later in the same function.	Required	Yes
Rule 15.4	There should be no more than one break or goto statement used to terminate any iteration statement.	Advisory	Yes
Rule 16.1	All switch statements shall be well-formed.	Required	Yes
Rule 16.4	Every switch statement shall have a default label.	Required	Yes
Rule 16.6	Every switch statement shall have at least two switch-clauses.	Required	Yes
Rule 16.7	A switch-expression shall not have essentially Boolean type.	Required	Yes
Rule 17.1	The features of <stdarg.h> shall not be used.	Required	Yes
Rule 18.4	The +, -, += and -= operators should not be applied to an expression of pointer type.	Advisory	Yes
Rule 18.7	Flexible array members shall not be declared.	Required	Yes
Rule 19.2	The union keyword should not be used.	Advisory	Yes
Rule 20.2	The ', " or \ characters and the /* or // character sequences shall not occur in a header file name.	Required	Yes
Rule 20.3	The #include directive shall be followed by either a <filename> or "filename" sequence.	Required	Yes
Rule 20.4	A macro shall not be defined with the same name as a keyword.	Required	Yes
Rule 20.5	#undef should not be used.	Advisory	Yes
Rule 20.8	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1.	Required	Yes
Rule 20.10	The # and ## preprocessor operators should not be used.	Advisory	Yes
Rule 20.11	A macro parameter immediately following a # operator shall not immediately be followed by a ## operator.	Required	Yes
Rule 20.12	A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators.	Required	Yes
Rule 21.1	#define and #undef shall not be used on a reserved identifier or reserved macro name.	Required	Yes
Rule 21.2	A reserved identifi er or macro name shall not be declared.	Required	Yes
Rule 21.3	The memory allocation and deallocation functions of <stdlib.h> shall not be used.	Required	Yes
Rule 21.4	The standard header file <setjmp.h> shall not be used.	Required	Yes
Rule 21.5	The standard header file <signal.h> shall not be used.	Required	Yes
Rule 21.6	The Standard Library input/output functions shall not be used.	Required	Yes
Rule 21.7	The atof, atoi, atol and atoll functions of <stdlib.h> shall not be used.	Required	Yes

Rule 21.8	The library functions abort, exit, getenv and system of <stdlib.h> shall not be used.	Required	Yes
Rule 21.9	The library functions bsearch and qsort of <stdlib.h> shall not be used.	Required	Yes
Rule 21.10	The Standard Library time and date functions shall not be used.	Required	Yes
Rule 21.11	The standard header file <tgmath.h> shall not be used.	Required	Yes
Rule 21.12	The exception handling features of <fenv.h> should not be used.	Required	Yes
Rule 22.1	All resources obtained dynamically by means of Standard Library functions shall be explicitly released.	Required	Yes Partial
Rule 22.2	A block of memory shall only be freed if it was allocated by means of a Standard Library function.	Required	Yes Partial