C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C code

1. "memset" should not be used to delete sensitive data
   Vulnerability
2. POSIX functions should not be called with arguments that trigger buffer overflows
   Vulnerability
3. XML parsers should not be vulnerable to XXE attacks
   Vulnerability
4. Function-like macros should not be invoked without all of their arguments
   Bug
5. The address of an automatic object should not be assigned to another object that may persist after the first object has ceased to exist
   Bug
6. "pthread_mutex_t" should be unlocked in the reverse order they were locked
   Bug
7. "pthread_mutex_t" should be properly initialized and destroyed
   Bug
8. "pthread_mutex_t" should not be consecutively locked or unlocked twice
   Bug
9. Functions with "noreturn" attribute should not return
   Bug
10. "memcmp" should only be called with pointers to trivially copyable types with no padding
    Bug
11. Stack allocated memory and non-owned memory should not be freed
    Bug
12. Closed resources should not be accessed
    Bug
13. Dynamically allocated memory should be released
    Bug
14. Freed memory should not be used
    Bug
15. Memory locations should not be released more than once
    Bug

16. Memory access should be explicitly bounded to prevent buffer overflows
Bug
17. Printf-style format strings should not lead to unexpected behavior at runtime
Bug
18. Recursion should not be infinite
Bug
19. Resources should be closed
Bug
20. Hard-coded credentials are security-sensitive
Security Hotspot
21. "goto" should jump to labels declared later in the same function
Code Smell
22. Only standard forms of the "defined" directive should be used
Code Smell
23. Switch labels should not be nested inside non-switch blocks
Code Smell
24. The right-hand operands of && and || should not contain side effects
Code Smell
25. Digraphs should not be used
Code Smell
26. Trigraphs should not be used
Code Smell
27. "case" ranges should cover multiple values
Code Smell
28. Array indices should be placed between brackets
Code Smell
29. Redundant pointer operator sequences should be removed
Code Smell
30. Non-reentrant POSIX functions should be replaced with their reentrant versions
Code Smell
31. "goto" statements should not be used to jump into blocks
Code Smell
32. Keywords introduced in later specifications should not be used as identifiers
Code Smell

33. Switch cases should end with an unconditional "break" statement
Code Smell

34. "switch" statements should not contain non-case labels
Code Smell

35. Control should not be transferred into a complex logic block using a "goto" or a "switch" statement
Code Smell

36. Accessing files should not introduce TOCTOU vulnerabilities
Vulnerability

37. Cipher algorithms should be robust
Vulnerability

38. Encryption algorithms should be used with secure mode and padding scheme
Vulnerability

39. Server hostnames should be verified during SSL/TLS connections
Vulnerability

40. Server certificates should be verified during SSL/TLS connections
Vulnerability

41. Cryptographic keys should be robust
Vulnerability

42. Weak SSL/TLS protocols should not be used
Vulnerability

43. Insecure functions should not be used
Vulnerability

44. "scanf()" and "fscanf()" format strings should specify a field width for the "%s" string placeholder
Vulnerability

45. Function exit paths should have appropriate return values
Bug

46. "volatile" should not be used to qualify objects for which the meaning is not defined
Bug

47. Relational and subtraction operators should not be used with pointers to different arrays
Bug

48. Arguments evaluation order should not be relied on
Bug

49.

	Parameter values should be appropriate Bug
50.	
	Zero should not be a possible denominator Bug
51.	
	Line-splicing should not be used in "//" comments Bug
52.	
	Pointers should not be cast to integral types Bug
53.	
	"sprintf" should not be used Security Hotspot
54.	
	Changing working directories without verifying the success is security-sensitive Security Hotspot
55.	
	Using "tmpnam", "tmpnam_s" or "tmpnam_r" is security-sensitive Security Hotspot
56.	
	Changing directories improperly when using "chroot" is security-sensitive Security Hotspot
57.	
	Using publicly writable directories is security-sensitive Security Hotspot
58.	
	Using clear-text protocols is security-sensitive Security Hotspot
59.	
	Expanding archive files without controlling resource consumption is security-sensitive Security Hotspot
60.	
	Using weak hashing algorithms is security-sensitive Security Hotspot
61.	
	Using pseudorandom number generators (PRNGs) is security-sensitive Security Hotspot
62.	
	"#undef" should be used with caution Code Smell
63.	
	Function names should be used either as a call with a parameter list or with the "&" operator Code Smell
64.	
	Functions should not be defined with a variable number of arguments Code Smell
65.	
	A cast shall not remove any const or volatile qualification from the type of a pointer or reference Code Smell

66. Object and function types should be explicitly stated in their declarations and definitions
Code Smell

67. Functions should be declared explicitly
Code Smell

68. Appropriate arguments should be passed to UNIX/POSIX functions
Code Smell

69. Appropriate arguments should be passed to stream functions
Code Smell

70. Blocking functions should not be called inside critical sections
Code Smell

71. Return value of "setuid" family of functions should always be checked
Code Smell

72. Size of variable length arrays should be positive
Code Smell

73. Argument of "printf" should be a format string
Code Smell

74. "mktemp" family of functions templates should have at least six trailing "X"s
Code Smell

75. Logical operators should not be confused with bitwise operators
Code Smell

76. Header guards should be followed by according "#define" macro
Code Smell

77. "default" clauses should be first or last
Code Smell

78. A conditionally executed single line should be denoted by indentation
Code Smell

79. Conditionals should start on new lines
Code Smell

80. Cognitive Complexity of functions should not be too high
Code Smell

81. Control characters should not be used in literals
Code Smell

82. "restrict" should not be used
Code Smell

83.	"static" should not be used for the size of an array parameter Code Smell
84.	The sign of an unsigned variable should not be tested Code Smell
85.	Pre-defined macros should not be defined, redefined or undefined Code Smell
86.	Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply Code Smell
87.	Methods should not be empty Code Smell
88.	Account validity should be verified when authenticating users with PAM Vulnerability
89.	Lines starting with "#" should contain valid preprocessing directives Bug
90.	"#include" directives should be followed by either <filename> or "filename" sequences Bug
91.	Non-standard characters should not occur in header file names in "#include" directives Bug
92.	Non-empty statements should change control flow or have at least one side-effect Bug
93.	Unary minus should not be applied to an unsigned expression Bug
94.	Objects with integer type should not be converted to objects with pointer type Bug
95.	Variables should be initialized before use Bug
96.	String literals with different prefixes should not be concatenated Bug
97.	Only escape sequences defined in the ISO C standard should be used Bug
98.	"#pragma pack" should be used correctly Bug
99.	Enums should be consistent with the bit fields they initialize

Bug

100. Array values should not be replaced unconditionally
Bug

101. Integral operations should not overflow
Bug

102. "case" ranges should not be empty
Bug

103. All branches in a conditional structure should not have exactly the same implementation
Bug

104. Declaration specifiers should not be redundant
Bug

105. "sizeof" should not be called on pointers
Bug

106. Unary prefix operators should not be repeated
Bug

107. "=+" should not be used instead of "+="
Bug

108. Values of different "enum" types should not be compared
Bug

109. Conditionally executed code should be reachable
Bug

110. Null pointers should not be dereferenced
Bug

111. Single-bit named bit fields should not be of a signed type
Bug

112. Values should not be uselessly incremented
Bug

113. "sizeof(sizeof(...))" should not be used
Bug

114. Related "if/else if" statements should not have the same condition
Bug

115. Identical expressions should not be used on both sides of a binary operator
Bug

116. All code should be reachable

Bug

117.

Loops with at most one iteration should be refactored
Bug

118.

Variables should not be self-assigned
Bug

119.

Setting capabilities is security-sensitive
Security Hotspot

120.

Using "strncpy" or "wcsncpy" is security-sensitive
Security Hotspot

121.

Using "strncat" or "wcsncat" is security-sensitive
Security Hotspot

122.

Using "strcat" or "wcscat" is security-sensitive
Security Hotspot

123.

Using "strlen" or "wcslen" is security-sensitive
Security Hotspot

124.

Using "strcpy" or "wcscpy" is security-sensitive
Security Hotspot

125.

Setting loose POSIX file permissions is security-sensitive
Security Hotspot

126.

#include directives in a file should only be preceded by other preprocessor directives or comments
Code Smell

127.

Loops should not have more than one "break" or "goto" statement
Code Smell

128.

Unused type declarations should be removed
Code Smell

129.

Comma operator should not be used
Code Smell

130.

"bool" expressions should not be used as operands to built-in operators other than =, &&, ||, !, ==, !=, unary &, and the conditional operator
Code Smell

131.

"enum" members other than the first one should not be explicitly initialized unless all members are explicitly initialized
Code Smell

132.

If a function has internal linkage then all re-declarations shall include the static storage class specifer

Code Smell

133.

Functions should not be declared at block scope
Code Smell

134.

Bit fields should be declared with appropriate types
Code Smell

135.

Size of bit fields should not exceed the size of their types
Code Smell

136.

GNU attributes should be used correctly
Code Smell

137.

Unevaluated operands should not have side effects
Code Smell

138.

Size argument of memory functions should be consistent
Code Smell

139.

Implicit casts should not lower precision
Code Smell

140.

Appropriate size arguments should be passed to "strncat" and "strlcpy"
Code Smell

141.

Keywords shall not be used as macros identifiers
Code Smell

142.

Dereferenced null pointers should not be bound to references
Code Smell

143.

"else" statements should be clearly matched with an "if"
Code Smell

144.

Include directives should not rely on non-portable search strategy
Code Smell

145.

"#include" paths should be portable
Code Smell

146.

"#import" should not be used
Code Smell

147.

Atomic types should be used instead of "volatile" types
Code Smell

148.

"switch" statements should cover all cases
Code Smell

149.

Methods returns should not be invariant

Code Smell

150.

Printf-style format strings should be used correctly
Code Smell

151.

Conditional operators should not be nested
Code Smell

152.

Multiline blocks should be enclosed in curly braces
Code Smell

153.

Increment should not be used to set boolean variables to 'true'
Code Smell

154.

Boolean expressions should not be gratuitous
Code Smell

155.

Parameters should be passed in the correct order
Code Smell

156.

Obsolete POSIX functions should not be used
Code Smell

157.

Two branches in a conditional structure should not have exactly the same implementation
Code Smell

158.

Unused assignments should be removed
Code Smell

159.

Structures should not have too many fields
Code Smell

160.

"switch" statements should not have too many "case" clauses
Code Smell

161.

Sections of code should not be commented out
Code Smell

162.

Deprecated K&R syntax should not be used for function definition
Code Smell

163.

Unused function parameters should be removed
Code Smell

164.

Unused functions and methods should be removed
Code Smell

165.

Track uses of "FIXME" tags
Code Smell

166.

	Deprecated attributes should include explanations Code Smell
167.	
	Assignments should not be made from within sub-expressions Code Smell
168.	
	Variables should not be shadowed Code Smell
169.	
	Redundant pairs of parentheses should be removed Code Smell
170.	
	Nested blocks of code should not be left empty Code Smell
171.	
	Functions should not have too many parameters Code Smell
172.	
	Collapsible "if" statements should be merged Code Smell
173.	
	Unused labels should be removed Code Smell
174.	
	The "sizeof" and "alignof" operator should not be used with operands of a "void" type Bug
175.	
	"nonnull" pointers should not be set to null Bug
176.	
	"for" loop counters should not have essentially floating type Bug
177.	
	Line continuation characters '\' should not be followed by trailing whitespace Bug
178.	
	Using hardcoded IP addresses is security-sensitive Security Hotspot
179.	
	Pointer and reference parameters should be "const" if the corresponding object is not modified Code Smell
180.	
	The three expressions of a "for" statement should only be concerned with loop control Code Smell
181.	
	Literal suffix "L" for long integers shall be upper case Code Smell
182.	
	Multicharacter literals should not be used Code Smell

183. Loop variables should be declared in the minimal possible scope
Code Smell

184. Macros should not be used as replacement to "typdef" and "using"
Code Smell

185. "^" should not be confused with exponentiation
Code Smell

186. Pointer and reference local variables should be "const" if the corresponding object is not modified
Code Smell

187. Format strings should comply with ISO standards
Code Smell

188. Functions which do not return should be declared as "noreturn"
Code Smell

189. Macros should not be redefined
Code Smell

190. "#include_next" should not be used
Code Smell

191. String literals should not be concatenated implicitly
Code Smell

192. Types and variables should be declared in separate statements
Code Smell

193. Jump statements should not be redundant
Code Smell

194. Empty "case" clauses that fall through to the "default" should be omitted
Code Smell

195. Forward declarations should not be redundant
Code Smell

196. Declarations should not be empty
Code Smell

197. Redundant casts should not be used
Code Smell

198. Code annotated as deprecated should not be used
Code Smell

199. "#pragma warning (default: ...)" should not be used

	Code Smell
200.	
	Init-declarator-lists and member-declarator-lists should consist of single init-declarators and member-declarators respectively Code Smell
201.	
	Unused local variables should be removed Code Smell
202.	
	"switch" statements should have at least 3 "case" clauses Code Smell
203.	
	A "while" loop should be used instead of a "for" loop Code Smell
204.	
	Nested code blocks should not be used Code Smell
205.	
	Empty statements should be removed Code Smell
206.	
	"/*" and "//" should not be used within comments Code Smell
207.	
	Track uses of "TODO" tags Code Smell
208.	
	Deprecated code should be removed Code Smell
209.	
	Reserved identifiers and functions in the C standard library should not be defined or declared Code Smell
210.	
	Bit fields should not be used Code Smell
211.	
	Track lack of copyright and license headers Code Smell
212.	
	Octal values should not be used Code Smell
213.	
	"abort", "exit", "getenv" and "system" from <stdlib.h> should not be used Bug
214.	
	"atof", "atoi" and "atol" from <stdlib.h> should not be used Bug
215.	
	"<signal.h>" should not be used Bug

216.
Dynamic heap memory allocation should not be used
Bug
217.
"<time.h>" should not be used
Code Smell
218.
"<stdio.h>" should not be used in production code
Code Smell
219.
"offsetof" macro from <stddef.h> should not be used
Code Smell
220.
"errno" should not be used
Code Smell
221.
"setjmp" and "longjmp" should not be used
Code Smell
222.
Function-like macros should not be used
Code Smell
223.
Macros should not be #define'd or #undef'd within a block
Code Smell
224.
Unions should not be used
Code Smell
225.
Object declarations should contain no more than 2 levels of pointer indirection
Code Smell
226.
Functions without parameters should be declared with parameter type "void"
Code Smell
227.
Recursion should not be used
Code Smell
228.
Constants of unsigned type should have a "U" suffix
Code Smell
229.
Octal and hexadecimal escape sequences should be terminated
Code Smell
230.
Flexible array members should not be declared
Code Smell
231.
Preprocessor directives should not be indented
Code Smell
232.
"switch" statements should not be nested
Code Smell

233. Cyclomatic Complexity of functions should not be too high
Code Smell

234. "switch" statements should have "default" clauses
Code Smell

235. "if ... else if" constructs should end with "else" clauses
Code Smell

236. "typedef" should be used for function pointers
Code Smell

237. Control structures should use curly braces
Code Smell

238. Expressions should not be too complex
Code Smell

239. Macros used in preprocessor directives should be defined before use
Bug

240. The number of arguments passed to a function should match the number of parameters
Bug

241. Evaluation of the operand to the sizeof operator shall not contain side effects
Bug

242. Bitwise operators should not be applied to signed operands
Bug

243. Boolean operations should not have numeric operands, and vice versa
Bug

244. Pointer conversions should be restricted to a safe subset
Bug

245. Function pointers should not be converted to any other type
Bug

246. Results of ~ and << operations on operands of underlying types unsigned char and unsigned short should immediately be cast to the operand's underlying type
Bug

247. User-defined types should not be passed as variadic arguments
Bug

248. The "<stdlib.h>" functions "bsearch" and "qsort" should not be used
Bug

249. Floating point numbers should not be tested for equality

Bug

250.

There shall be at most one occurrence of the # or ## operators in a single macro definition
Code Smell

251.

Parameters in a function prototype should be named
Code Smell

252.

"goto" statement should not be used
Code Smell

253.

Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression
Code Smell

254.

"enum" values should not be used as operands to built-in operators other than [ ], =, ==, !=, unary &, and the relational operators <, <=, >, >=
Code Smell

255.

Operands of "&&" and "||" should be primary (C) or postfix (C++) expressions
Code Smell

256.

Limited dependence should be placed on operator precedence
Code Smell

257.

The value of a complex expression should only be cast to a type that is narrower and of the same signedness as the underlying type of the expression
Code Smell

258.

Braces should be used to indicate and match the structure in the non-zero initialization of arrays and structures
Code Smell

259.

Array declarations should include an explicit size specification
Code Smell

260.

"typedef" names should be unique identifiers
Code Smell

261.

Identifiers should not be longer than 31 characters
Code Smell

262.

All uses of the #pragma directive should be documented
Code Smell

263.

Assembly language should be encapsulated and isolated
Code Smell

264.

Functions that are not used in a project should be removed
Code Smell

265.

	Track parsing failures Code Smell
266.	
	Files should not be too complex Code Smell
267.	
	The ternary operator should not be used Code Smell
268.	
	Functions/methods should not have too many lines Code Smell
269.	
	Track uses of "NOSONAR" comments Code Smell
270.	
	"for" loop stop conditions should be invariant Code Smell
271.	
	Statements should be on separate lines Code Smell
272.	
	"switch case" clauses should not have too many lines of code Code Smell
273.	
	Functions should not contain too many return statements Code Smell
274.	
	Magic numbers should not be used Code Smell
275.	
	Files should not have too many lines of code Code Smell
276.	
	Lines should not be too long Code Smell
277.	
	Function parameters' initial values should not be ignored Bug
278.	
	Preprocessor operators "#" and "##" should not be used Code Smell
279.	
	Structure and union types should be complete at the end of a translation unit Code Smell
280.	
	Switch statement conditions should not have essentially boolean type Code Smell
281.	
	"continue" should not be used Code Smell
282.	

	Tests of non-Boolean values against zero should be explicit Code Smell
283.	
	Signed and unsigned types should not be mixed in expressions Code Smell
284.	
	typedefs that indicate size and signedness should be used in place of the basic types Code Smell
285.	
	Appropriate char types should be used for character and integer values Code Smell
286.	
	Source code should only use /* ... */ style comments Code Smell
287.	
	GNU extensions should not be used Code Smell
288.	
	Methods should not return constants Code Smell
289.	
	Label names should comply with a naming convention Code Smell
290.	
	Enumeration values should comply with a naming convention Code Smell
291.	
	Enumeration names should comply with a naming convention Code Smell
292.	
	Comment styles "//" and "/* ... */" should not be mixed within a file Code Smell
293.	
	"union" names should comply with a naming convention Code Smell
294.	
	Constants should come first in equality tests Code Smell
295.	
	Type specifiers should be listed in a standard order Code Smell
296.	
	Track "TODO" and "FIXME" comments that do not contain a reference to a person Code Smell
297.	
	The prefix increment/decrement form should be used Code Smell
298.	
	"struct" names should comply with a naming convention Code Smell
299.	

	File names should comply with a naming convention Code Smell
300.	
	Macro names should comply with a naming convention Code Smell
301.	
	Comments should not be located at the end of lines of code Code Smell
302.	
	break statements should not be used except for switch cases Code Smell
303.	
	Local variable and function parameter names should comply with a naming convention Code Smell
304.	
	Field names should comply with a naming convention Code Smell
305.	
	Lines should not end with trailing whitespaces Code Smell
306.	
	Files should contain an empty newline at the end Code Smell
307.	
	Tabulation characters should not be used Code Smell
308.	
	A function should have a single point of exit at the end of the function Code Smell
309.	
	Function names should comply with a naming convention Code Smell
310.	
	Track comments matching a regular expression Code Smell
311.	
	Track instances of the "#error" preprocessor directive being reached Code Smell