Objective C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your OBJECTIVE 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. Function-like macros should not be invoked without all of their arguments
   Bug
4. 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
5. "pthread_mutex_t" should be unlocked in the reverse order they were locked
   Bug
6. "pthread_mutex_t" should be properly initialized and destroyed
   Bug
7. "pthread_mutex_t" should not be consecutively locked or unlocked twice
   Bug
8. Functions with "noreturn" attribute should not return
   Bug
9. "memcmp" should only be called with pointers to trivially copyable types with no padding
   Bug
10. Stack allocated memory and non-owned memory should not be freed
    Bug
11. Closed resources should not be accessed
    Bug
12. Dynamically allocated memory should be released
    Bug
13. Freed memory should not be used
    Bug
14. Memory locations should not be released more than once
    Bug
15. Memory access should be explicitly bounded to prevent buffer overflows
    Bug

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

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

34. Accessing files should not introduce TOCTOU vulnerabilities
Vulnerability

35. Cipher algorithms should be robust
Vulnerability

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

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

38. Cryptographic keys should be robust
Vulnerability

39. Insecure functions should not be used
Vulnerability

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

41. Function exit paths should have appropriate return values
Bug

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

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

44. Arguments evaluation order should not be relied on
Bug

45. Parameter values should be appropriate
Bug

46. Zero should not be a possible denominator
Bug

47. Line-splicing should not be used in "//" comments
Bug

48. Pointers should not be cast to integral types
Bug

49.

	"return" statements should not occur in "finally" blocks Bug
50.	
	"sprintf" should not be used Security Hotspot
51.	
	Changing working directories without verifying the success is security-sensitive Security Hotspot
52.	
	Using "tmpnam", "tmpnam_s" or "tmpnam_r" is security-sensitive Security Hotspot
53.	
	Changing directories improperly when using "chroot" is security-sensitive Security Hotspot
54.	
	Using publicly writable directories is security-sensitive Security Hotspot
55.	
	Using clear-text protocols is security-sensitive Security Hotspot
56.	
	Expanding archive files without controlling resource consumption is security-sensitive Security Hotspot
57.	
	Using weak hashing algorithms is security-sensitive Security Hotspot
58.	
	Using pseudorandom number generators (PRNGs) is security-sensitive Security Hotspot
59.	
	"#undef" should be used with caution Code Smell
60.	
	Function names should be used either as a call with a parameter list or with the "&" operator Code Smell
61.	
	Functions should not be defined with a variable number of arguments Code Smell
62.	
	A cast shall not remove any const or volatile qualification from the type of a pointer or reference Code Smell
63.	
	Object and function types should be explicitly stated in their declarations and definitions Code Smell
64.	
	Functions should be declared explicitly Code Smell
65.	
	Appropriate arguments should be passed to UNIX/POSIX functions Code Smell

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

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

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

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

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

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

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

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

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

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

76. Conditionals should start on new lines
Code Smell

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

78. Keywords should be used before arguments
Code Smell

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

80. "restrict" should not be used
Code Smell

81. "static" should not be used for the size of an array parameter
Code Smell

82. The sign of an unsigned variable should not be tested
Code Smell

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

Bug

100. Integral operations should not overflow
Bug

101. "case" ranges should not be empty
Bug

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

103. Declaration specifiers should not be redundant
Bug

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

105. Unary prefix operators should not be repeated
Bug

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

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

108. Null pointers should not be dereferenced
Bug

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

110. Values should not be uselessly incremented
Bug

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

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

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

114. All code should be reachable
Bug

115. Loops with at most one iteration should be refactored
Bug

116. Variables should not be self-assigned

Bug

117.

Condition-specific "catch" handlers should not be used after the ellipsis (catch-all) handler
Bug

118.

Setting capabilities is security-sensitive
Security Hotspot

119.

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

120.

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

121.

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

122.

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

123.

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

124.

Setting loose POSIX file permissions is security-sensitive
Security Hotspot

125.

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

126.

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

127.

Unused type declarations should be removed
Code Smell

128.

Comma operator should not be used
Code Smell

129.

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

130.

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

131.

Functions should not be declared at block scope
Code Smell

132.

Bit fields should be declared with appropriate types

Code Smell

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

134. GNU attributes should be used correctly
Code Smell

135. Unevaluated operands should not have side effects
Code Smell

136. Size argument of memory functions should be consistent
Code Smell

137. Implicit casts should not lower precision
Code Smell

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

139. Keywords shall not be used as macros identifiers
Code Smell

140. Dereferenced null pointers should not be bound to references
Code Smell

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

142. Methods should not have identical implementations
Code Smell

143. "#include" paths should be portable
Code Smell

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

145. "switch" statements should cover all cases
Code Smell

146. C declarations should not be made inside Objective-C structures
Code Smell

147. Printf-style format strings should be used correctly
Code Smell

148. Conditional operators should not be nested
Code Smell

149. Multiline blocks should be enclosed in curly braces

Code Smell

150.

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

151.

Parameters should be passed in the correct order
Code Smell

152.

Obsolete POSIX functions should not be used
Code Smell

153.

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

154.

Unused assignments should be removed
Code Smell

155.

Structures should not have too many fields
Code Smell

156.

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

157.

Classes should not have too many methods
Code Smell

158.

Sections of code should not be commented out
Code Smell

159.

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

160.

Unused function parameters should be removed
Code Smell

161.

Unused functions and methods should be removed
Code Smell

162.

Try-catch blocks should not be nested
Code Smell

163.

Track uses of "FIXME" tags
Code Smell

164.

Deprecated attributes should include explanations
Code Smell

165.

Assignments should not be made from within sub-expressions
Code Smell

166.

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

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

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

185. Format strings should comply with ISO standards
Code Smell

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

187. Macros should not be redefined
Code Smell

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

189. String literals should not be concatenated implicitly
Code Smell

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

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

192. Forward declarations should not be redundant
Code Smell

193. Declarations should not be empty
Code Smell

194. General "catch" clauses should not be used
Code Smell

195. "catch" clauses should do more than rethrow
Code Smell

196. Exceptions should not be ignored
Code Smell

197. Curly braces should not be used on interfaces without instance variables
Code Smell

198. Redundant casts should not be used
Code Smell

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

200.

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

201.

Init-declarator-lists and member-declarator-lists should consist of single init-declarators and member-declarators respectively
Code Smell

202.

Unused local variables should be removed
Code Smell

203.

"switch" statements should have at least 3 "case" clauses
Code Smell

204.

A "while" loop should be used instead of a "for" loop
Code Smell

205.

Nested code blocks should not be used
Code Smell

206.

Empty statements should be removed
Code Smell

207.

"/*" and "//" should not be used within comments
Code Smell

208.

Track uses of "TODO" tags
Code Smell

209.

Deprecated code should be removed
Code Smell

210.

Reserved identifiers and functions in the C standard library should not be defined or declared
Code Smell

211.

Bit fields should not be used
Code Smell

212.

Track lack of copyright and license headers
Code Smell

213.

Octal values should not be used
Code Smell

214.

"abort", "exit", "getenv" and "system" from <stdlib.h> should not be used
Bug

215.

"atof", "atoi" and "atol" from <stdlib.h> should not be used
Bug

216.

	"<signal.h>" should not be used Bug
217.	
	Dynamic heap memory allocation should not be used Bug
218.	
	"<time.h>" should not be used Code Smell
219.	
	"<stdio.h>" should not be used in production code Code Smell
220.	
	"offsetof" macro from <stddef.h> should not be used Code Smell
221.	
	"errno" should not be used Code Smell
222.	
	"setjmp" and "longjmp" should not be used Code Smell
223.	
	Function-like macros should not be used Code Smell
224.	
	Macros should not be #define'd or #undef'd within a block Code Smell
225.	
	Unions should not be used Code Smell
226.	
	Object declarations should contain no more than 2 levels of pointer indirection Code Smell
227.	
	Functions without parameters should be declared with parameter type "void" Code Smell
228.	
	Recursion should not be used Code Smell
229.	
	Constants of unsigned type should have a "U" suffix Code Smell
230.	
	Octal and hexadecimal escape sequences should be terminated Code Smell
231.	
	Flexible array members should not be declared Code Smell
232.	
	Preprocessor directives should not be indented Code Smell
233.	

	"switch" statements should not be nested Code Smell
234.	
	Cyclomatic Complexity of functions should not be too high Code Smell
235.	
	Cyclomatic Complexity of classes should not be too high Code Smell
236.	
	"switch" statements should have "default" clauses Code Smell
237.	
	"if ... else if" constructs should end with "else" clauses Code Smell
238.	
	"typedef" should be used for function pointers Code Smell
239.	
	Control structures should use curly braces Code Smell
240.	
	Expressions should not be too complex Code Smell
241.	
	Macros used in preprocessor directives should be defined before use Bug
242.	
	The number of arguments passed to a function should match the number of parameters Bug
243.	
	Evaluation of the operand to the sizeof operator shall not contain side effects Bug
244.	
	Bitwise operators should not be applied to signed operands Bug
245.	
	Boolean operations should not have numeric operands, and vice versa Bug
246.	
	Pointer conversions should be restricted to a safe subset Bug
247.	
	Function pointers should not be converted to any other type Bug
248.	
	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
249.	
	User-defined types should not be passed as variadic arguments Bug

250.

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

251.

Floating point numbers should not be tested for equality
Bug

252.

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

253.

Parameters in a function prototype should be named
Code Smell

254.

"goto" statement should not be used
Code Smell

255.

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

256.

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

257.

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

258.

Limited dependence should be placed on operator precedence
Code Smell

259.

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

260.

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

261.

Array declarations should include an explicit size specification
Code Smell

262.

"typedef" names should be unique identifiers
Code Smell

263.

Identifiers should not be longer than 31 characters
Code Smell

264.

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

265.

Assembly language should be encapsulated and isolated

Code Smell

266.

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

267.

Track parsing failures
Code Smell

268.

Files should not be too complex
Code Smell

269.

The ternary operator should not be used
Code Smell

270.

Exceptions should not be used
Code Smell

271.

Functions/methods should not have too many lines
Code Smell

272.

Track uses of "NOSONAR" comments
Code Smell

273.

"for" loop stop conditions should be invariant
Code Smell

274.

Statements should be on separate lines
Code Smell

275.

"switch case" clauses should not have too many lines of code
Code Smell

276.

Functions should not contain too many return statements
Code Smell

277.

Magic numbers should not be used
Code Smell

278.

Files should not have too many lines of code
Code Smell

279.

Lines should not be too long
Code Smell

280.

Function parameters' initial values should not be ignored
Bug

281.

Preprocessor operators "#" and "##" should not be used
Code Smell

282.

Structure and union types should be complete at the end of a translation unit

Code Smell

283.

Switch statement conditions should not have essentially boolean type
Code Smell

284.

"continue" should not be used
Code Smell

285.

Tests of non-Boolean values against zero should be explicit
Code Smell

286.

Signed and unsigned types should not be mixed in expressions
Code Smell

287.

typedefs that indicate size and signedness should be used in place of the basic types
Code Smell

288.

Appropriate char types should be used for character and integer values
Code Smell

289.

Source code should only use /* ... */ style comments
Code Smell

290.

GNU extensions should not be used
Code Smell

291.

Methods should not return constants
Code Smell

292.

Label names should comply with a naming convention
Code Smell

293.

Enumeration values should comply with a naming convention
Code Smell

294.

Enumeration names should comply with a naming convention
Code Smell

295.

Comment styles "//" and "/* ... */" should not be mixed within a file
Code Smell

296.

"union" names should comply with a naming convention
Code Smell

297.

Constants should come first in equality tests
Code Smell

298.

Type specifiers should be listed in a standard order
Code Smell

299.

Track "TODO" and "FIXME" comments that do not contain a reference to a person

Code Smell

300.

The prefix increment/decrement form should be used
Code Smell

301.

"struct" names should comply with a naming convention
Code Smell

302.

File names should comply with a naming convention
Code Smell

303.

Macro names should comply with a naming convention
Code Smell

304.

Comments should not be located at the end of lines of code
Code Smell

305.

break statements should not be used except for switch cases
Code Smell

306.

Local variable and function parameter names should comply with a naming convention
Code Smell

307.

Field names should comply with a naming convention
Code Smell

308.

Lines should not end with trailing whitespaces
Code Smell

309.

Files should contain an empty newline at the end
Code Smell

310.

Tabulation characters should not be used
Code Smell

311.

Class names should comply with a naming convention
Code Smell

312.

A function should have a single point of exit at the end of the function
Code Smell

313.

Function names should comply with a naming convention
Code Smell

314.

Track comments matching a regular expression
Code Smell

315.

Track instances of the "#error" preprocessor directive being reached
Code Smell