C# static code analysis

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

1. HTTP responses should not be vulnerable to session fixation
   Vulnerability
2. Extracting archives should not lead to zip slip vulnerabilities
   Vulnerability
3. Dynamic code execution should not be vulnerable to injection attacks
   Vulnerability
4. HTTP request redirections should not be open to forging attacks
   Vulnerability
5. Deserialization should not be vulnerable to injection attacks
   Vulnerability
6. Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks
   Vulnerability
7. "CoSetProxyBlanket" and "CoInitializeSecurity" should not be used
   Vulnerability
8. Database queries should not be vulnerable to injection attacks
   Vulnerability
9. XML parsers should not be vulnerable to XXE attacks
   Vulnerability
10. A secure password should be used when connecting to a database
    Vulnerability
11. XPath expressions should not be vulnerable to injection attacks
    Vulnerability
12. I/O function calls should not be vulnerable to path injection attacks
    Vulnerability
13. LDAP queries should not be vulnerable to injection attacks
    Vulnerability
14. OS commands should not be vulnerable to command injection attacks
    Vulnerability
15. Classes should implement their "ExportAttribute" interfaces
    Bug
16.

	Neither "Thread.Resume" nor "Thread.Suspend" should be used Bug
17.	
	"SafeHandle.DangerousGetHandle" should not be called Bug
18.	
	Type inheritance should not be recursive Bug
19.	
	"IDisposables" should be disposed Bug
20.	
	SQL keywords should be delimited by whitespace Bug
21.	
	Composite format strings should not lead to unexpected behavior at runtime Bug
22.	
	Recursion should not be infinite Bug
23.	
	Destructors should not throw exceptions Bug
24.	
	Hard-coded credentials are security-sensitive Security Hotspot
25.	
	Exceptions should not be thrown from unexpected methods Code Smell
26.	
	"operator==" should not be overloaded on reference types Code Smell
27.	
	Type should not be examined on "System.Type" instances Code Smell
28.	
	Test method signatures should be correct Code Smell
29.	
	Method overloads with default parameter values should not overlap Code Smell
30.	
	"value" parameters should be used Code Smell
31.	
	"is" should not be used with "this" Code Smell
32.	
	Methods named "Dispose" should implement "IDisposable.Dispose" Code Smell
33.	

	Tests should include assertions Code Smell
34.	
	Silly bit operations should not be performed Code Smell
35.	
	Public methods should not have multidimensional array parameters Code Smell
36.	
	"async" and "await" should not be used as identifiers Code Smell
37.	
	TestCases should contain tests Code Smell
38.	
	Short-circuit logic should be used in boolean contexts Code Smell
39.	
	JWT should be signed and verified with strong cipher algorithms Vulnerability
40.	
	Cipher algorithms should be robust Vulnerability
41.	
	Encryption algorithms should be used with secure mode and padding scheme Vulnerability
42.	
	Insecure temporary file creation methods should not be used Vulnerability
43.	
	Server certificates should be verified during SSL/TLS connections Vulnerability
44.	
	LDAP connections should be authenticated Vulnerability
45.	
	Cryptographic keys should be robust Vulnerability
46.	
	Weak SSL/TLS protocols should not be used Vulnerability
47.	
	Cipher Block Chaining IVs should be unpredictable Vulnerability
48.	
	Regular expressions should not be vulnerable to Denial of Service attacks Vulnerability
49.	
	Hashes should include an unpredictable salt Vulnerability
50.	

	Non-async "Task/Task<T>" methods should not return null Bug
51.	
	Calls to delegate's method "BeginInvoke" should be paired with calls to "EndInvoke" Bug
52.	
	"Shared" parts should not be created with "new" Bug
53.	
	Getters and setters should access the expected fields Bug
54.	
	Right operands of shift operators should be integers Bug
55.	
	Shared resources should not be used for locking Bug
56.	
	Locks should be released Bug
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.	
	Configuring loggers is security-sensitive Security Hotspot
61.	
	Using weak hashing algorithms is security-sensitive Security Hotspot
62.	
	Disabling CSRF protections is security-sensitive Security Hotspot
63.	
	Using non-standard cryptographic algorithms is security-sensitive Security Hotspot
64.	
	Using pseudorandom number generators (PRNGs) is security-sensitive Security Hotspot
65.	
	Parameter names should match base declaration and other partial definitions Code Smell
66.	
	"ValueTask" should be consumed correctly Code Smell
67.	

	String offset-based methods should be preferred for finding substrings from offsets Code Smell
68.	
	"default" clauses should be first or last Code Smell
69.	
	Unread "private" fields should be removed Code Smell
70.	
	Base class methods should not be hidden Code Smell
71.	
	Inherited member visibility should not be decreased Code Smell
72.	
	Threads should not lock on objects with weak identity Code Smell
73.	
	A conditionally executed single line should be denoted by indentation Code Smell
74.	
	Conditionals should start on new lines Code Smell
75.	
	Assemblies should have version information Code Smell
76.	
	Exception types should be "public" Code Smell
77.	
	Cognitive Complexity of methods should not be too high Code Smell
78.	
	"params" should not be introduced on overrides Code Smell
79.	
	"[DefaultValue]" should not be used when "[DefaultParameterValue]" is meant Code Smell
80.	
	"[Optional]" should not be used on "ref" or "out" parameters Code Smell
81.	
	Non-flags enums should not be used in bitwise operations Code Smell
82.	
	Inner class members should not shadow outer class "static" or type members Code Smell
83.	
	"Explicit" conversions of "foreach" loops should not be used Code Smell
84.	

	Instance members should not write to "static" fields Code Smell
85.	
	"IndexOf" checks should not be for positive numbers Code Smell
86.	
	Whitespace and control characters in string literals should be explicit Code Smell
87.	
	Properties should not make collection or array copies Code Smell
88.	
	Flags enumerations zero-value members should be named "None" Code Smell
89.	
	Overflow checking should not be disabled for "Enumerable.Sum" Code Smell
90.	
	Field-like events should not be virtual Code Smell
91.	
	Non-constant static fields should not be visible Code Smell
92.	
	Inappropriate casts should not be made Code Smell
93.	
	Constructors should only call non-overridable methods Code Smell
94.	
	"GC.Collect" should not be called Code Smell
95.	
	Methods should not be empty Code Smell
96.	
	Exceptions should not be thrown in finally blocks Code Smell
97.	
	Method overrides should not change parameter defaults Code Smell
98.	
	Types allowed to be deserialized should be restricted Vulnerability
99.	
	Server-side requests should not be vulnerable to forging attacks Vulnerability
100.	
	Members should not have conflicting transparency annotations Vulnerability
101.	

"PartCreationPolicyAttribute" should be used with "ExportAttribute"
Bug

102.

"ConstructorArgument" parameters should exist in constructors
Bug

103.

Windows Forms entry points should be marked with STAThread
Bug

104.

Collection elements should not be replaced unconditionally
Bug

105.

Exceptions should not be created without being thrown
Bug

106.

Collection sizes and array length comparisons should make sense
Bug

107.

Serialization event handlers should be implemented correctly
Bug

108.

Deserialization methods should be provided for "OptionalField" members
Bug

109.

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

110.

Types should be defined in named namespaces
Bug

111.

Empty nullable value should not be accessed
Bug

112.

Nullable type comparison should not be redundant
Bug

113.

Methods with "Pure" attribute should return a value
Bug

114.

One-way "OperationContract" methods should have "void" return type
Bug

115.

Optional parameters should be passed to "base" calls
Bug

116.

Classes should not have only "private" constructors
Bug

117.

Expressions used in "Debug.Assert" should not produce side effects
Bug

118.

	Caller information parameters should come at the end of the parameter list Bug
119.	
	Static fields should appear in the order they must be initialized Bug
120.	
	Classes directly extending "object" should not call "base" in "GetHashCode" or "Equals" Bug
121.	
	Anonymous delegates should not be used to unsubscribe from Events Bug
122.	
	Delegates should not be subtracted Bug
123.	
	"async" methods should not return "void" Bug
124.	
	"ThreadStatic" should not be used on non-static fields Bug
125.	
	"IDisposables" created in a "using" statement should not be returned Bug
126.	
	"ThreadStatic" fields should not be initialized Bug
127.	
	"Object.ReferenceEquals" should not be used for value types Bug
128.	
	Doubled prefix operators "!!" and "~~" should not be used Bug
129.	
	"=+" should not be used instead of "+=" Bug
130.	
	"NaN" should not be used in comparisons Bug
131.	
	Conditionally executed code should be reachable Bug
132.	
	Null pointers should not be dereferenced Bug
133.	
	For-loop conditions should be true at least once Bug
134.	
	A "for" loop update clause should move the counter in the right direction Bug
135.	

"ToString()" method should not return null
Bug

136.

Return values from functions without side effects should not be ignored
Bug

137.

Values should not be uselessly incremented
Bug

138.

Collections should not be passed as arguments to their own methods
Bug

139.

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

140.

Objects should not be created to be dropped immediately without being used
Bug

141.

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

142.

Loops with at most one iteration should be refactored
Bug

143.

Variables should not be self-assigned
Bug

144.

Constructing arguments of system commands from user input is security-sensitive
Security Hotspot

145.

Deserializing objects without performing data validation is security-sensitive
Security Hotspot

146.

Disabling ASP.NET "Request Validation" feature is security-sensitive
Security Hotspot

147.

Allowing requests with excessive content length is security-sensitive
Security Hotspot

148.

Setting loose file permissions is security-sensitive
Security Hotspot

149.

Formatting SQL queries is security-sensitive
Security Hotspot

150.

Using hardcoded IP addresses is security-sensitive
Security Hotspot

151.

"goto" statement should not be used
Code Smell

152.

	"new Guid()" should not be used Code Smell
153.	
	Parameter validation in "async"/"await" methods should be wrapped Code Smell
154.	
	Parameter validation in yielding methods should be wrapped Code Smell
155.	
	Events should have proper arguments Code Smell
156.	
	"P/Invoke" methods should not be visible Code Smell
157.	
	Native methods should be wrapped Code Smell
158.	
	Methods should not have identical implementations Code Smell
159.	
	Non-flags enums should not be marked with "FlagsAttribute" Code Smell
160.	
	Classes implementing "IEquatable<T>" should be sealed Code Smell
161.	
	"GC.SuppressFinalize" should not be called Code Smell
162.	
	Objects should not be disposed more than once Code Smell
163.	
	Parameter names used into ArgumentException constructors should match an existing one Code Smell
164.	
	"ISerializable" should be implemented correctly Code Smell
165.	
	"Assembly.Load" should be used Code Smell
166.	
	"IDisposable" should be implemented correctly Code Smell
167.	
	"ServiceContract" and "OperationContract" attributes should be used together Code Smell
168.	
	Composite format strings should be used correctly Code Smell

169. Exceptions should not be explicitly rethrown
Code Smell

170. "abstract" classes should not have "public" constructors
Code Smell

171. Assertion arguments should be passed in the correct order
Code Smell

172. Ternary operators should not be nested
Code Smell

173. Events should be invoked
Code Smell

174. "params" should be used on overrides
Code Smell

175. Generic type parameters should be co/contravariant when possible
Code Smell

176. Multiple "OrderBy" calls should not be used
Code Smell

177. Reflection should not be used to increase accessibility of classes, methods, or fields
Code Smell

178. Static fields should not be updated in constructors
Code Smell

179. "IEnumerable" LINQs should be simplified
Code Smell

180. Fields that are only assigned in the constructor should be "readonly"
Code Smell

181. Static fields should not be used in generic types
Code Smell

182. Multiline blocks should be enclosed in curly braces
Code Smell

183. Boolean expressions should not be gratuitous
Code Smell

184. Types and methods should not have too many generic parameters
Code Smell

185. Write-only properties should not be used
Code Smell

186. Exceptions should not be thrown from property getters
Code Smell

187. Unused type parameters should be removed
Code Smell

188. Parameters should be passed in the correct order
Code Smell

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

190. Unused assignments should be removed
Code Smell

191. Tests should not be ignored
Code Smell

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

193. Sections of code should not be commented out
Code Smell

194. Unused method parameters should be removed
Code Smell

195. Empty arrays and collections should be returned instead of null
Code Smell

196. Unused private types or members should be removed
Code Smell

197. Track uses of "FIXME" tags
Code Smell

198. "Obsolete" attributes should include explanations
Code Smell

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

200. General exceptions should never be thrown
Code Smell

201. Utility classes should not have public constructors
Code Smell

202. Local variables should not shadow class fields

Code Smell

203.

Redundant pairs of parentheses should be removed
Code Smell

204.

Inheritance tree of classes should not be too deep
Code Smell

205.

Nested blocks of code should not be left empty
Code Smell

206.

Methods should not have too many parameters
Code Smell

207.

Collapsible "if" statements should be merged
Code Smell

208.

OS commands should not be vulnerable to argument injection attacks
Vulnerability

209.

Logging should not be vulnerable to injection attacks
Vulnerability

210.

Empty collections should not be accessed or iterated
Bug

211.

Mutable, non-private fields should not be "readonly"
Bug

212.

"string.ToCharArray()" should not be called redundantly
Bug

213.

"base.Equals" should not be used to check for reference equality in "Equals" if "base" is not "object"
Bug

214.

Property assignments should not be made for "readonly" fields not constrained to reference types
Bug

215.

Flags enumerations should explicitly initialize all their members
Bug

216.

"GetHashCode" should not reference mutable fields
Bug

217.

Results of integer division should not be assigned to floating point variables
Bug

218.

Integral numbers should not be shifted by zero or more than their number of bits-1
Bug

219.	"Equals(Object)" and "GetHashCode()" should be overridden in pairs Bug
220.	Having a permissive Cross-Origin Resource Sharing policy is security-sensitive Security Hotspot
221.	Delivering code in production with debug features activated is security-sensitive Security Hotspot
222.	Searching OS commands in PATH is security-sensitive Security Hotspot
223.	Creating cookies without the "HttpOnly" flag is security-sensitive Security Hotspot
224.	Creating cookies without the "secure" flag is security-sensitive Security Hotspot
225.	Literal suffixes should be upper case Code Smell
226.	Null checks should not be used with "is" Code Smell
227.	Method overloads should be grouped together Code Smell
228.	"params" should be used instead of "varargs" Code Smell
229.	"static" fields should be initialized inline Code Smell
230.	Classes that provide "Equals(<T>)" should implement "IEquatable<T>" Code Smell
231.	Jump statements should not be redundant Code Smell
232.	Member initializer values should not be redundant Code Smell
233.	Unassigned members should be removed Code Smell
234.	Empty "case" clauses that fall through to the "default" should be omitted Code Smell
235.	Parameters with "[DefaultParameterValue]" attributes should also be marked "[Optional]" Code Smell

236.

Interfaces should not simply inherit from base interfaces with colliding members
Code Smell

237.

Variables should not be checked against the values they're about to be assigned
Code Smell

238.

Methods should not return constants
Code Smell

239.

Attribute, EventArgs, and Exception type names should end with the type being extended
Code Smell

240.

Loops should be simplified with "LINQ" expressions
Code Smell

241.

Namespaces should not be empty
Code Smell

242.

Non-derived "private" classes and records should be "sealed"
Code Smell

243.

"string.IsNullOrEmpty" should be used
Code Smell

244.

Implementations should be provided for "partial" methods
Code Smell

245.

Duplicate casts should not be made
Code Smell

246.

Methods should not return values that are never used
Code Smell

247.

Caller information arguments should not be provided explicitly
Code Smell

248.

Method calls should not resolve ambiguously to overloads with "params"
Code Smell

249.

"catch" clauses should do more than rethrow
Code Smell

250.

Generic exceptions should not be ignored
Code Smell

251.

Mutable fields should not be "public static"
Code Smell

252.

Enumeration type names should not have "Flags" or "Enum" suffixes
Code Smell

253.
Enumeration types should comply with a naming convention
Code Smell
254.
Trivial properties should be auto-implemented
Code Smell
255.
Runtime type checking should be simplified
Code Smell
256.
Boolean checks should not be inverted
Code Smell
257.
Inheritance list should not be redundant
Code Smell
258.
Redundant casts should not be used
Code Smell
259.
Strings should not be concatenated using '+' in a loop
Code Smell
260.
Unused local variables should be removed
Code Smell
261.
Private fields only used as local variables in methods should become local variables
Code Smell
262.
A "while" loop should be used instead of a "for" loop
Code Smell
263.
"Equals" and the comparison operators should be overridden when implementing "IComparable"
Code Smell
264.
Nested code blocks should not be used
Code Smell
265.
Overriding members should do more than simply call the same member in the base class
Code Smell
266.
"Any()" should be used to test for emptiness
Code Smell
267.
Boolean literals should not be redundant
Code Smell
268.
Empty statements should be removed
Code Smell
269.
Fields should not have public accessibility

Code Smell

270.

URIs should not be hardcoded
Code Smell

271.

Types should be named in PascalCase
Code Smell

272.

Track uses of "TODO" tags
Code Smell

273.

Classes with "IDisposable" members should implement "IDisposable"
Bug

274.

Calls to "async" methods should not be blocking
Code Smell

275.

Child class fields should not shadow parent class fields
Code Smell

276.

Track lack of copyright and license headers
Code Smell

277.

Exit methods should not be called
Code Smell

278.

Classes should "Dispose" of members from the classes' own "Dispose" methods
Bug

279.

Reading the Standard Input is security-sensitive
Security Hotspot

280.

Using command line arguments is security-sensitive
Security Hotspot

281.

Using Sockets is security-sensitive
Security Hotspot

282.

Encrypting data is security-sensitive
Security Hotspot

283.

Using regular expressions is security-sensitive
Security Hotspot

284.

Interface methods should be callable by derived types
Code Smell

285.

Child class fields should not differ from parent class fields only by capitalization
Code Smell

286.

Pointers to unmanaged memory should not be visible

Code Smell

287.

Number patterns should be regular
Code Smell

288.

"out" and "ref" parameters should not be used
Code Smell

289.

Unchanged local variables should be "const"
Code Smell

290.

"ConfigureAwait(false)" should be used
Code Smell

291.

"interface" instances should not be cast to concrete types
Code Smell

292.

Literal boolean values should not be used in assertions
Code Smell

293.

Optional parameters should not be used
Code Smell

294.

Public constant members should not be used
Code Smell

295.

Array covariance should not be used
Code Smell

296.

"nameof" should be used
Code Smell

297.

Modulus results should not be checked for direct equality
Code Smell

298.

"for" loop increment clauses should modify the loops' counters
Code Smell

299.

"switch" statements should not be nested
Code Smell

300.

Methods and properties should not be too complex
Code Smell

301.

Control flow statements "if", "switch", "for", "foreach", "while", "do" and "try" should not be nested too deeply
Code Smell

302.

"switch/Select" statements should contain a "default/Case Else" clauses
Code Smell

303.

	"if ... else if" constructs should end with "else" clauses Code Smell
304.	
	Control structures should use curly braces Code Smell
305.	
	Expressions should not be too complex Code Smell
306.	
	ASP.NET HTTP request validation feature should not be disabled Vulnerability
307.	
	Serialization constructors should be secured Vulnerability
308.	
	Calculations should not overflow Bug
309.	
	Floating point numbers should not be tested for equality Bug
310.	
	Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression Code Smell
311.	
	Use a testable date/time provider. Code Smell
312.	
	Property names should not match get methods Code Smell
313.	
	Locales should be set for data types Code Smell
314.	
	Literals should not be passed as localized parameters Code Smell
315.	
	Operators should be overloaded consistently Code Smell
316.	
	Method signatures should not contain nested generic types Code Smell
317.	
	Enumeration members should not be named "Reserved" Code Smell
318.	
	"System.Uri" arguments should be used instead of strings Code Smell
319.	
	Collection properties should be readonly Code Smell

320. Disposable types should declare finalizers
Code Smell

321. String URI overloads should call "System.Uri" overloads
Code Smell

322. URI properties should not be strings
Code Smell

323. URI return values should not be strings
Code Smell

324. URI Parameters should not be strings
Code Smell

325. Custom attributes should be marked with "System.AttributeUsageAttribute"
Code Smell

326. Assemblies should explicitly specify COM visibility
Code Smell

327. Assemblies should be marked as CLS compliant
Code Smell

328. "Generic.List" instances should not be part of public APIs
Code Smell

329. Collections should implement the generic interface
Code Smell

330. Generic event handlers should be used
Code Smell

331. Event Handlers should have the correct signature
Code Smell

332. "Assembly.GetExecutingAssembly" should not be called
Code Smell

333. Arguments of public methods should be validated against null
Code Smell

334. Value types should implement "IEquatable<T>"
Code Smell

335. Finalizers should not be empty
Code Smell

336. "[ExpectedException]" should not be used
Code Smell

337.

"this" should not be exposed from constructors
Code Smell

338.

Types should not have members with visibility set higher than the type's visibility
Code Smell

339.

Fields should be private
Code Smell

340.

"try" statements with identical "catch" and/or "finally" blocks should be merged
Code Smell

341.

NullReferenceException should not be caught
Code Smell

342.

Functions should not have too many lines of code
Code Smell

343.

"for" loop stop conditions should be invariant
Code Smell

344.

Statements should be on separate lines
Code Smell

345.

Classes should not be coupled to too many other classes (Single Responsibility Principle)
Code Smell

346.

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

347.

Magic numbers should not be used
Code Smell

348.

Standard outputs should not be used directly to log anything
Code Smell

349.

Files should not have too many lines of code
Code Smell

350.

Lines should not be too long
Code Smell

351.

HTTP response headers should not be vulnerable to injection attacks
Vulnerability

352.

Console logging should not be used
Vulnerability

353.

Generic parameters not constrained to reference types should not be compared to "null"
Bug

354.

The length returned from a stream read should be checked
Bug

355.

Method parameters, caught exceptions and foreach variables' initial values should not be ignored
Bug

356.

Controlling permissions is security-sensitive
Security Hotspot

357.

Writing cookies is security-sensitive
Security Hotspot

358.

Methods should be named according to their synchronicities
Code Smell

359.

Extensions should be in separate namespaces
Code Smell

360.

Extension methods should not extend "object"
Code Smell

361.

Operator overloads should have named alternatives
Code Smell

362.

Non-abstract attributes should be sealed
Code Smell

363.

Overloads with a "StringComparison" parameter should be used
Code Smell

364.

Overloads with a "CultureInfo" or an "IFormatProvider" parameter should be used
Code Smell

365.

Types should not extend outdated base types
Code Smell

366.

Properties should be preferred
Code Smell

367.

Generics should be used when appropriate
Code Smell

368.

Type names should not match namespaces
Code Smell

369.

Strings should be normalized to uppercase
Code Smell

370.

Exceptions should provide standard constructors

Code Smell

371.

Assemblies should be marked with "NeutralResourcesLanguageAttribute"
Code Smell

372.

Interfaces should not be empty
Code Smell

373.

Enumerations should have "Int32" storage
Code Smell

374.

Generic methods should provide type parameters
Code Smell

375.

Multidimensional arrays should not be used
Code Smell

376.

"static readonly" constants should be "const" instead
Code Smell

377.

Strings or integral types should be used for indexers
Code Smell

378.

Parameter names should not duplicate the names of their methods
Code Smell

379.

Track use of "NotImplementedException"
Code Smell

380.

Empty "default" clauses should be removed
Code Smell

381.

Redundant property names should be omitted in anonymous classes
Code Smell

382.

Declarations and initializations should be as concise as possible
Code Smell

383.

Default parameter values should not be passed as arguments
Code Smell

384.

Constructor and destructor declarations should not be redundant
Code Smell

385.

Method parameters should be declared with base types
Code Smell

386.

The simplest possible condition syntax should be used
Code Smell

387.

Redundant parentheses should not be used

Code Smell

388.

"GC.SuppressFinalize" should not be invoked for types without destructors
Code Smell

389.

Members should not be initialized to default values
Code Smell

390.

Sequential tests should not check the same condition
Code Smell

391.

Redundant modifiers should not be used
Code Smell

392.

Methods and properties that don't access instance data should be static
Code Smell

393.

"Exception" should not be caught when not required by called methods
Code Smell

394.

"sealed" classes should not have "protected" members
Code Smell

395.

Underscores should be used to make large numbers readable
Code Smell

396.

"ToString()" calls should not be redundant
Code Smell

397.

"==" should not be used when "Equals" is overridden
Code Smell

398.

An abstract class should have both abstract and concrete methods
Code Smell

399.

Multiple variables should not be declared on the same line
Code Smell

400.

Culture should be specified for "string" operations
Code Smell

401.

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

402.

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

403.

String literals should not be duplicated
Code Smell

404.

Files should contain an empty newline at the end

	Code Smell
405.	
	Unused "using" should be removed Code Smell
406.	
	A close curly brace should be located at the beginning of a line Code Smell
407.	
	Tabulation characters should not be used Code Smell
408.	
	Methods and properties should be named in PascalCase Code Smell
409.	
	Track uses of in-source issue suppressions Code Smell