Python static code analysis

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

1. HTTP responses should not be vulnerable to session fixation
   Vulnerability
2. Dynamic code execution should not be vulnerable to injection attacks
   Vulnerability
3. NoSQL operations 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. Database queries should not be vulnerable to injection attacks
   Vulnerability
8. XML parsers should not be vulnerable to XXE attacks
   Vulnerability
9. A secure password should be used when connecting to a database
   Vulnerability
10. XPath expressions should not be vulnerable to injection attacks
    Vulnerability
11. I/O function calls should not be vulnerable to path injection attacks
    Vulnerability
12. LDAP queries should not be vulnerable to injection attacks
    Vulnerability
13. OS commands should not be vulnerable to command injection attacks
    Vulnerability
14. The number and name of arguments passed to a function should match its parameters
    Bug
15. The "open" builtin function should be called with a valid mode
    Bug
16.

Only defined names should be listed in "__all__"
Bug

17.

Calls should not be made to non-callable values
Bug

18.

Property getter, setter and deleter methods should have the expected number of parameters
Bug

19.

Special methods should have an expected number of parameters
Bug

20.

Instance and class methods should have at least one positional parameter
Bug

21.

Boolean expressions of exceptions should not be used in "except" statements
Bug

22.

Caught Exceptions must derive from BaseException
Bug

23.

Item operations should be done on objects supporting them
Bug

24.

Raised Exceptions must derive from BaseException
Bug

25.

Operators should be used on compatible types
Bug

26.

Function arguments should be passed only once
Bug

27.

Iterable unpacking, "for-in" loops and "yield from" should use an Iterable object
Bug

28.

Variables, classes and functions should be defined before being used
Bug

29.

Identity operators should not be used with dissimilar types
Bug

30.

Only strings should be listed in "__all__"
Bug

31.

"__init__" should not return a value
Bug

32.

"yield" and "return" should not be used outside functions
Bug

33.	String formatting should not lead to runtime errors	Bug
34.	Recursion should not be infinite	Bug
35.	Silly equality checks should not be made	Bug
36.	Granting access to S3 buckets to all or authenticated users is security-sensitive	Security Hotspot
37.	Hard-coded credentials are security-sensitive	Security Hotspot
38.	Functions returns should not be invariant	Code Smell
39.	The "exec" statement should not be used	Code Smell
40.	Backticks should not be used	Code Smell
41.	Methods and field names should not differ only by capitalization	Code Smell
42.	JWT should be signed and verified	Vulnerability
43.	Cipher algorithms should be robust	Vulnerability
44.	Encryption algorithms should be used with secure mode and padding scheme	Vulnerability
45.	Server hostnames should be verified during SSL/TLS connections	Vulnerability
46.	Insecure temporary file creation methods should not be used	Vulnerability
47.	Server certificates should be verified during SSL/TLS connections	Vulnerability
48.	LDAP connections should be authenticated	Vulnerability
49.	Cryptographic key generation should be based on strong parameters	Vulnerability

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

51. Cipher Block Chaining IVs should be unpredictable
Vulnerability

52. Regular expressions should not be vulnerable to Denial of Service attacks
Vulnerability

53. Hashes should include an unpredictable salt
Vulnerability

54. Regex lookahead assertions should not be contradictory
Bug

55. Regex boundaries should not be used in a way that can never be matched
Bug

56. Exceptions' "__cause__" should be either an Exception or None
Bug

57. "break" and "continue" should not be used outside a loop
Bug

58. Break, continue and return statements should not occur in "finally" blocks
Bug

59. Allowing public ACLs or policies on a S3 bucket is security-sensitive
Security Hotspot

60. Using publicly writable directories is security-sensitive
Security Hotspot

61. Using clear-text protocols is security-sensitive
Security Hotspot

62. Expanding archive files without controlling resource consumption is security-sensitive
Security Hotspot

63. Signalling processes is security-sensitive
Security Hotspot

64. Configuring loggers is security-sensitive
Security Hotspot

65. Using weak hashing algorithms is security-sensitive
Security Hotspot

66. Disabling CSRF protections is security-sensitive
Security Hotspot

67.

Using non-standard cryptographic algorithms is security-sensitive
Security Hotspot

68.

Using pseudorandom number generators (PRNGs) is security-sensitive
Security Hotspot

69.

Constants should not be used as conditions
Code Smell

70.

"SystemExit" should be re-raised
Code Smell

71.

Bare "raise" statements should only be used in "except" blocks
Code Smell

72.

Comparison to None should not be constant
Code Smell

73.

"self" should be the first argument to instance methods
Code Smell

74.

Function parameters' default values should not be modified or assigned
Code Smell

75.

Some special methods should return "NotImplemented" instead of raising "NotImplementedError"
Code Smell

76.

Custom Exception classes should inherit from "Exception" or one of its subclasses
Code Smell

77.

Bare "raise" statements should not be used in "finally" blocks
Code Smell

78.

Arguments given to functions should be of an expected type
Code Smell

79.

`str.replace` should be preferred to `re.sub`
Code Smell

80.

Unread "private" attributes should be removed
Code Smell

81.

Cognitive Complexity of functions should not be too high
Code Smell

82.

The first argument to class methods should follow the naming convention
Code Smell

83.

Method overrides should not change contracts

Code Smell

84.

Wildcard imports should not be used
Code Smell

85.

String literals should not be duplicated
Code Smell

86.

Functions and methods should not be empty
Code Smell

87.

Server-side requests should not be vulnerable to forging attacks
Vulnerability

88.

Non-empty statements should change control flow or have at least one side-effect
Bug

89.

Replacement strings should reference existing regular expression groups
Bug

90.

Alternation in regular expressions should not contain empty alternatives
Bug

91.

Unicode Grapheme Clusters should be avoided inside regex character classes
Bug

92.

Regex alternatives should not be redundant
Bug

93.

Alternatives in regular expressions should be grouped when used with anchors
Bug

94.

New objects should not be created only to check their identity
Bug

95.

Collection content should not be replaced unconditionally
Bug

96.

Exceptions should not be created without being raised
Bug

97.

Collection sizes and array length comparisons should make sense
Bug

98.

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

99.

The output of functions that don't return anything should not be used
Bug

100.

"=+" should not be used instead of "+="

Bug

101.

Increment and decrement operators should not be used
Bug

102.

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

103.

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

104.

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

105.

All code should be reachable
Bug

106.

Loops with at most one iteration should be refactored
Bug

107.

Variables should not be self-assigned
Bug

108.

All "except" blocks should be able to catch exceptions
Bug

109.

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

110.

Disabling auto-escaping in template engines is security-sensitive
Security Hotspot

111.

Setting loose POSIX file permissions is security-sensitive
Security Hotspot

112.

Formatting SQL queries is security-sensitive
Security Hotspot

113.

Character classes in regular expressions should not contain only one character
Code Smell

114.

Superfluous curly brace quantifiers should be avoided
Code Smell

115.

Non-capturing groups without quantifier should not be used
Code Smell

116.

Regular expressions should not contain empty groups
Code Smell

117.

Regular expressions should not contain multiple spaces

Code Smell

118.

Single-character alternations in regular expressions should be replaced with character classes
Code Smell

119.

Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string
Code Smell

120.

Values assigned to variables should match their type annotations
Code Smell

121.

Function return types should be consistent with their type hint
Code Smell

122.

Character classes in regular expressions should not contain the same character twice
Code Smell

123.

Type checks shouldn't be confusing
Code Smell

124.

Regular expressions should not be too complicated
Code Smell

125.

Builtins should not be shadowed by local variables
Code Smell

126.

Implicit string and byte concatenations should not be confusing
Code Smell

127.

Identity comparisons should not be used with cached typed
Code Smell

128.

Expressions creating sets should not have duplicate values
Code Smell

129.

Expressions creating dictionaries should not have duplicate keys
Code Smell

130.

Special method "__exit__" should not re-raise the provided exception
Code Smell

131.

Unused scope-limited definitions should be removed
Code Smell

132.

Functions and methods should not have identical implementations
Code Smell

133.

Unused private nested classes should be removed
Code Smell

134.

String formatting should be used correctly
Code Smell

135.

Conditional expressions should not be nested
Code Smell

136.

Loops without "break" should not have "else" clauses
Code Smell

137.

Doubled prefix operators "not" and "~" should not be used
Code Smell

138.

The "print" statement should not be used
Code Smell

139.

"<>" should not be used to test inequality
Code Smell

140.

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

141.

Unused assignments should be removed
Code Smell

142.

A field should not duplicate the name of its containing class
Code Smell

143.

Function names should comply with a naming convention
Code Smell

144.

Functions and lambdas should not reference variables defined in enclosing loops
Code Smell

145.

Sections of code should not be commented out
Code Smell

146.

Unused function parameters should be removed
Code Smell

147.

Unused class-private methods should be removed
Code Smell

148.

Track uses of "FIXME" tags
Code Smell

149.

"Exception" and "BaseException" should not be raised
Code Smell

150.

Redundant pairs of parentheses should be removed

Code Smell

151.

Nested blocks of code should not be left empty
Code Smell

152.

Functions, methods and lambdas should not have too many parameters
Code Smell

153.

Collapsible "if" statements should be merged
Code Smell

154.

Logging should not be vulnerable to injection attacks
Vulnerability

155.

Repeated patterns in regular expressions should not match the empty string
Bug

156.

Function parameters initial values should not be ignored
Bug

157.

Disabling versioning of S3 buckets is security-sensitive
Security Hotspot

158.

Disabling server-side encryption of S3 buckets is security-sensitive
Security Hotspot

159.

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive
Security Hotspot

160.

Delivering code in production with debug features activated is security-sensitive
Security Hotspot

161.

Allowing both safe and unsafe HTTP methods is security-sensitive
Security Hotspot

162.

Creating cookies without the "HttpOnly" flag is security-sensitive
Security Hotspot

163.

Creating cookies without the "secure" flag is security-sensitive
Security Hotspot

164.

Using hardcoded IP addresses is security-sensitive
Security Hotspot

165.

Regular expression quantifiers and character classes should be used concisely
Code Smell

166.

Character classes should be preferred over reluctant quantifiers in regular expressions
Code Smell

167.

A subclass should not be in the same "except" statement as a parent class

		Code Smell
168.	Walrus operator should not make code confusing	Code Smell
169.	Jump statements should not be redundant	Code Smell
170.	"pass" should not be used needlessly	Code Smell
171.	"except" clauses should do more than raise the same issue	Code Smell
172.	Boolean checks should not be inverted	Code Smell
173.	Unused local variables should be removed	Code Smell
174.	Local variable and function parameter names should comply with a naming convention	Code Smell
175.	Field names should comply with a naming convention	Code Smell
176.	Class names should comply with a naming convention	Code Smell
177.	Method names should comply with a naming convention	Code Smell
178.	Track uses of "TODO" tags	Code Smell
179.	HTML autoescape mechanism should not be globally disabled	Vulnerability
180.	Variables, classes and functions should be either defined or imported	Bug
181.	"__exit__" should accept type, value, and traceback arguments	Bug
182.	"return" and "yield" should not be used in the same function	Bug
183.	Track lack of copyright and license headers	Code Smell
184.	HTTP response headers should not be vulnerable to injection attacks	

Vulnerability

185.
Regular expressions should be syntactically valid
Bug

186.
Sending emails is security-sensitive
Security Hotspot

187.
Reading the Standard Input is security-sensitive
Security Hotspot

188.
Using command line arguments is security-sensitive
Security Hotspot

189.
Encrypting data is security-sensitive
Security Hotspot

190.
Using regular expressions is security-sensitive
Security Hotspot

191.
Dynamically executing code is security-sensitive
Security Hotspot

192.
Cyclomatic Complexity of functions should not be too high
Code Smell

193.
Control flow statements "if", "for", "while", "try" and "with" should not be nested too deeply
Code Smell

194.
Cyclomatic Complexity of classes should not be too high
Code Smell

195.
"\" should only be used as an escape character outside of raw strings
Bug

196.
Using shell interpreter when executing OS commands is security-sensitive
Security Hotspot

197.
Functions should use "return" consistently
Code Smell

198.
Python parser failure
Code Smell

199.
Files should not be too complex
Code Smell

200.
Docstrings should be defined
Code Smell

201.
Functions should not have too many lines of code

Code Smell

202.

Track uses of "NOSONAR" comments
Code Smell

203.

Track comments matching a regular expression
Code Smell

204.

Statements should be on separate lines
Code Smell

205.

Functions should not contain too many return statements
Code Smell

206.

Files should not have too many lines of code
Code Smell

207.

Lines should not be too long
Code Smell

208.

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

209.

New-style classes should be used
Code Smell

210.

Parentheses should not be used after certain keywords
Code Smell

211.

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

212.

Module names should comply with a naming convention
Code Smell

213.

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

214.

Lines should not end with trailing whitespaces
Code Smell

215.

Files should contain an empty newline at the end
Code Smell

216.

Long suffix "L" should be upper case
Code Smell