Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

Python static code analysis

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

All rules 216 | Vulnerability 29 | Bug 55 | Security Hotspot 31 | Code Smell 101

Tags

Search by name...

Silly equality checks should not be made
Bug

Granting access to S3 buckets to all or authenticated users is security-sensitive
Security Hotspot

Hard-coded credentials are security-sensitive
Security Hotspot

Functions returns should not be invariant
Code Smell

The "exec" statement should not be used
Code Smell

Backticks should not be used
Code Smell

Methods and field names should not differ only by capitalization
Code Smell

JWT should be signed and verified
Vulnerability

Cipher algorithms should be robust
Vulnerability

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

Server hostnames should be verified during SSL/TLS connections
Vulnerability

Instance and class methods should have at least one positional parameter

Analyze your code

Bug Blocker

Every instance method is expected to have at least one positional parameter. This parameter will reference the object instance on which the method is called. Calling an instance method which doesn't have at least one parameter will raise a "TypeError". By convention, this first parameter is usually named "self".

Class methods, i.e. methods annotated with `@classmethod`, also require at least one parameter. The only differences is that it will receive the class itself instead of a class instance. By convention, this first parameter is usually named "cls". Note that `__new__` and `__init_subclass__` take a class as first argument even thought they are not decorated with `@classmethod`.

This rule raises an issue when an instance of class method does not have at least one positional parameter.

Noncompliant Code Example

```
class MyClass:
    def instance_method():  # Noncompliant. "self" para
        print("instance_method")

    @classmethod
    def class_method():  # Noncompliant. "cls" paramete
        print("class_method")
```

Compliant Solution

```
class MyClass:
    def instance_method(self):
        print("instance_method")

    @classmethod
    def class_method(cls):
        print("class_method")


    @staticmethod
    def static_method():
        print("static_method")
```

See

- Python documentation - Method Objects

Available In:

Insecure temporary file creation methods should not be used

Vulnerability

Server certificates should be verified during SSL/TLS connections

Vulnerability

LDAP connections should be authenticated

Vulnerability

Cryptographic key generation should be based on strong parameters

Vulnerability

Weak SSL/TLS protocols should not

sonarlint | sonarcloud | sonarqube