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...

String formatting should not lead to runtime errors
Bug

Recursion should not be infinite
Bug

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

Special methods should have an expected number of parameters

Analyze your code

Bug Blocker

Python developers can customize how code is interpreted by defining special methods (also called magic methods). For example, it is possible to override how the multiplication operator (`a * b`) will apply to instances of a class by defining in this class the `__mul__` and `__rmul__` methods. Whenever a multiplication operation is performed with this class, the python interpreter will call one of these methods instead of performing the default multiplication.

The python interpreter will always call these methods with the same number of parameters. Every call to a special method will fail if it is defined with an unexpected number of parameters.

This rule raises an issue when a special method is defined with an unexpected number of parameters.

Noncompliant Code Example

```
class A:
    def __mul__(self, other, unexpected):  # Noncomplia
        return 42

    def __add__(self):  # Noncompliant. Missing one par
        return 42

A() * 3  # TypeError: __mul__() missing 1 required posi
A() + 3  # TypeError: __add__() takes 1 positional argu
```

Compliant Solution

```
class A:
    def __mul__(self, other):
        return 42

    def __add__(self, other):
        return 42

A() * 3
A() + 3
```

See

- Python Documentation - Special method names
- Python Documentation - copy module

Available In:

with secure mode and padding scheme

Vulnerability

Server hostnames should be verified during SSL/TLS connections

Vulnerability

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

sonarlint | sonarcloud | sonarqube