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...

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

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

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

Analyze your code

Bug Blocker

Property getter, setter and deleter methods are called by the python interpreter with a specific number or arguments:

- only "self" for property getter and deleter methods.
- "self" and a value for setter methods.

Adding any other parameter, or removing these mandatory parameters will make method calls fail.

This rule raises an issue when:

- too many parameters are defined in a property getter, setter or deleter method.
- the value parameter is missing in a property setter method.

Noncompliant Code Example

```
class A:
    @property
    def foo(self, unexpected, unexpected2):  # Noncompl
        return self._foo

    @foo.setter
    def foo(self, value, unexpected):  # Noncompliant.
        self._foo = value

    @foo.deleter
    def foo(self, unexpected):  # Noncompliant. Too man
        del self._foo

class B:
    def get_foo(self, unexpected):  # Noncompliant. Too
        return self._foo

    def set_foo(self, value, unexpected):  # Noncomplia
        self._foo = value

    def del_foo(self, unexpected):  # Noncompliant. Too
        del self._foo

    foo = property(get_foo, set_foo, del_foo, "'foo' pr
```

Compliant Solution

```
class A:
    @property
    def foo(self):
```

Vulnerability

Encryption algorithms should be used 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

```
        return self._foo

    @foo.setter
    def foo(self, value):
        self._foo = value

    @foo.deleter
    def foo(self):
        del self._foo

class B:
    def get_foo(self):
        return self._foo

    def set_foo(self, value):
        self._foo = value

    def del_foo(self):
        del self._foo

    foo = property(get_foo, set_foo, del_foo, "'foo' pr
```

See

- Python Documentation - Built-in Functions - property

Available In:

sonarlint | sonarcloud | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
Privacy Policy