

- 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



String formatting should not lead to runtime errors



Recursion should not be infinite



Silly equality checks should not be made



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



Hard-coded credentials are security-sensitive



Functions returns should not be invariant



The "exec" statement should not be used



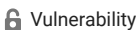
Backticks should not be used



Methods and field names should not differ only by capitalization



JWT should be signed and verified



Cipher algorithms should be robust

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

Analyze your code



Property getter, setter and deleter methods are called by the python interpreter with a specific number of 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): # Noncompliant
        return self._foo

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

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

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






    def set_foo(self, value, unexpected): # Noncompliant. Too many parameters
        self._foo = value

    def del_foo(self, unexpected): # Noncompliant. Too many parameters
        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:

 |  | 