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...

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

LDAP connections should be authenticated
Vulnerability

Cryptographic key generation should be based on strong parameters
Vulnerability

Weak SSL/TLS protocols should not be used
Vulnerability

Cipher Block Chaining IVs should be unpredictable
Vulnerability

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

Hashes should include an unpredictable salt
Vulnerability

Regex lookahead assertions should not be contradictory

Item operations should be done on objects supporting them

Analyze your code

Bug Blocker

Getting, setting and deleting items using square brackets requires the accessed object to have special methods:

- Getting items such as `my_variable[key]` requires `my_variable` to have the `__getitem__` method, or the `__class_getitem__` method if `my_variable` is a class.
- Setting items such as `my_variable[key] = 42` requires `my_variable` to have the `__setitem__` method.
- Deleting items such as `del my_variable[key]` requires `my_variable` to have the `__delitem__` method.

This rule raises an issue when an item operation is performed on an object which doesn't have the corresponding method.

Noncompliant Code Example

```
del (1, 2)[0]  # Noncompliant, tuples are immutable
(1, 2)[0] = 42  # Noncompliant
(1, 2)[0]

class A:
    def __init__(self, values):
        self._values = values

a = A([0,1,2])

a[0]  # Noncompliant
del a[0]  # Noncompliant
a[0] = 42  # Noncompliant

class B:
    pass

B[0]  # Noncompliant
```

Compliant Solution

```
del [1, 2][0]  # Lists are mutable
[1, 2][0] = 42
[1, 2][0]

class A:
    def __init__(self, values):
        self._values = values

    def __getitem__(self, key):
        return self._values[key]
```

Bug

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

Bug

Exceptions' "__cause__" should be either an Exception or None

Bug

"break" and "continue" should not be used outside a loop

Bug

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

Bug

```
    def __setitem__(self, key, value):
        self._values[key] = value

    def __delitem__(self, key):
        del self._values[key]

a = A([0,1,2])

a[0]
del a[0]
a[0] = 42

class B:
    def __class_getitem__(cls, key):
        return [0, 1, 2, 3][key]

B[0]
```

See

- Python documentation - __getitem__ method
- Python documentation - __setitem__ method
- Python documentation - __delitem__ method
- Python documentation - __class_getitem__ method

Available In:

sonarlint | sonarcloud | sonarqube