

- 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...



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

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

Boolean expressions of exceptions should not be used in "except" statements

Analyze your code

Bug Blocker

The only two possible types for an except's expression are a class deriving from `BaseException`, or a tuple composed of such classes (or an old style class if you are using python 2, but this has been removed in python 3).

This rule raises an issue when the expression used in an except block is a boolean expression of exceptions. The result of such expression is a single exception class, which is valid but not what the developer intended.

Noncompliant Code Example

```
try:
    raise TypeError()
except ValueError or TypeError: # Noncompliant
    print("Catching only ValueError")
except ValueError and TypeError: # Noncompliant
    print("catching only TypeError")
except (ValueError or TypeError) as exception: # Noncompliant
    print("Catching only ValueError")

foo = ValueError or TypeError # foo == ValueError
foo = ValueError and TypeError # foo == TypeError
```

Compliant Solution

```
try:
    raise TypeError()
except (ValueError, TypeError) as exception:
    print("Catching all exceptions")
```


See

- Python documentation - [the try statement](#)


Available In:

sonarlint | sonarcloud | sonarqube


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

Randomized locations should