sonar RULES

Products

- 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...

injection attacks

Vulnerability

Repeated patterns in regular expressions should not match the empty string

Bug

Function parameters initial values should not be ignored

Bug

Disabling versioning of S3 buckets is security-sensitive

Security Hotspot

Disabling server-side encryption of S3 buckets is security-sensitive

Security Hotspot

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

Security Hotspot

Delivering code in production with debug features activated is security-sensitive

Security Hotspot

Allowing both safe and unsafe HTTP methods is security-sensitive

Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive

Security Hotspot

Creating cookies without the "secure" flag is security-sensitive

Security Hotspot

Using hardcoded IP addresses is security-sensitive

Security Hotspot

Regular expression quantifiers and character classes should be used

Functions and methods should not be empty

Analyze your code

Code Smell | Critical | suspicious

There are several reasons for a function or a method not to have a body:

- It is an unintentional omission, and should be fixed to prevent an unexpected behavior in production.
- It is not yet, or never will be, supported. In this case an exception should be thrown.
- The method is an intentionally-blank override. In this case a nested comment should explain the reason for the blank override.

Noncompliant Code Example

```
def myfunc1(foo="Noncompliant"):
    pass

class MyClass:
    def mymethod1(self, foo="Noncompliant"):
        pass
```

Compliant Solution

```
def myfunc1():
    pass  # comment explaining why this function is empty

def myfunc2():
    raise NotImplementedError()

def myfunc3():
    """
    Docstring explaining why this function is empty.
    """

class MyClass:
    def mymethod1(self):
        pass  # comment explaining why this function is empt

    def mymethod2(self):
        raise NotImplementedError()

    def mymethod3(self):
        """
        Docstring explaining why this method is empty. Note
        which are meant to be subclassed.
        """
```

Exceptions

No issue will be raised when the empty method is abstract and meant to be overriden in a subclass, i.e. it is decorated with `abc.abstractmethod`, `abc.abstractstaticmethod`, `abc.abstractclassmethod` or concisely

Code Smell

Character classes should be preferred over reluctant quantifiers in regular expressions

Code Smell

A subclass should not be in the same "except" statement as a parent class

Code Smell

Walrus operator should not make code confusing

Code Smell

`abc.abstractproperty`. Note however that these methods should normally have a docstring explaining how subclasses should implement these methods.

```
import abc

class MyAbstractClass(abc.ABC):
    @abc.abstractproperty
    def myproperty(self):
        pass

    @abc.abstractclassmethod
    def myclassmethod(cls):
        pass

    @abc.abstractmethod
    def mymethod(self):
        pass

    @abc.abstractstaticmethod
    def mystaticmethod():
        pass
```

Available In:

sonarlint | sonarcloud | sonarqube