



ABAP

Apex

C С

(C++

CloudFormation

COBOL

C#

3 CSS

 \bowtie Flex

-GO Go

5 HTML

Java

JavaScript

Kotlin

Objective C

PHP OiD.

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) 6 Vulnerability 29

∰ Bug (55)

Security Hotspot 31

Code Smell (101)

Search by name...

HTTP response headers should not be vulnerable to injection attacks

Vulnerability

Regular expressions should be syntactically valid

👬 Bug

Sending emails is security-sensitive

Security Hotspot

Reading the Standard Input is security-sensitive

Security Hotspot

Using command line arguments is security-sensitive

Security Hotspot

Encrypting data is security-sensitive

Security Hotspot

Using regular expressions is securitysensitive

Security Hotspot

Dynamically executing code is security-sensitive

Security Hotspot

Cyclomatic Complexity of functions should not be too high

A Code Smell

Control flow statements "if", "for", "while", "try" and "with" should not be nested too deeply

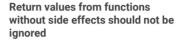
Code Smell

Cyclomatic Complexity of classes should not be too high

Code Smell

"\" should only be used as an escape character outside of raw strings

₩ Bug



Analyze your code

Tags

🛊 Bug 🔷 Major 🕝

When the call to a function doesn't have any side effects, what is the point of making the call if the results are ignored? In such case, either the function call is useless and should be dropped or the source code doesn't behave as expected.

This rule raises an issue when a builtin function or methods which has no side effects is called and its result is not used.

Noncompliant Code Example

```
myvar = "this is a multiline"
"message from {}".format(sender) # Noncompliant. The format
```

Compliant Solution

```
myvar = ("this is a multiline"
"message from {}".format(sender))
```

Exceptions

No issue will be raised when the function or method call is in a try...except body. This usually indicates that an exception is expected, and this exception is the side-effect.

```
def tryExcept():
   d = \{\}
    try:
       d[1]
    except IndexError as e:
        pass
        divmod(1, 0)
    except ZeroDivisionError as e:
        pass
```

See

• Python documentation - Built-in Functions

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

Using shell interpreter when executing OS commands is security-sensitive

Security Hotspot

Functions should use "return" consistently

Code Smell

Python parser failure

Code Smell

Files should not be too complex

Code Smell

SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy