



Secrets



Apex

C С



CloudFormation









-GO Go

5 HTML



JavaScript JS

Kotlin

Objective C

PHP

PL/I

PL/SQL



RPG

Ruby

Scala

D Swift

Terraform

Text

тѕ **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



6 Vulnerability (29)



Security Hotspot 31



Code Smell (101)

Tags

Search by name...

compatible types

Operators should be used on

Analyze your code







Calling an operator in python is equivalent to calling a special method (except for the identity operator is). Python provides a set of built-in operations. It is for example possible to add two integers: 1 + 2. It is however not possible to add a string and an integer: 1 $\,$ + $\,$ "2" and such an operation will raise a TypeError.

It is possible to define how an operator will behave with a custom class by defining the corresponding special method. See python documentation for a complete list of operators and their methods: arithmetic and bitwise operators, comparison operators.

For symmetrical binary operators you need to define two methods so that the order of operands doesn't matter, ex: add and radd

This rule raises an issue when an operator is used on incompatible types. Types are considered incompatible if no built-in operations between those types exist and none of the operands has implemented the corresponding special methods.

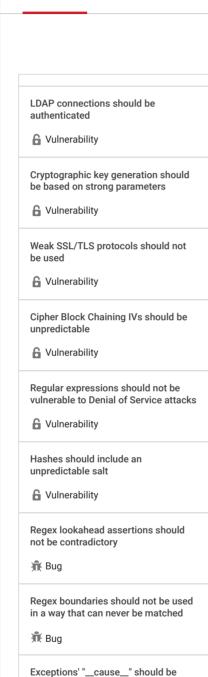
Noncompliant Code Example

```
class Empty:
   pass
class Add:
   def __add__(self, other):
        return 42
1 + "2" # Noncompliant
Empty() + 1 # Noncompliant
Add() + 1
1 + Add() # Noncompliant
Add() + Emptv()
Empty() + Add() # Noncompliant
```

Compliant Solution

```
class Empty:
class Add:
    def __add__(self, other):
        return 42
    def __radd__(self, other):
        return 42
```





either an Exception or None

used outside a loop

"break" and "continue" should not be

Break, continue and return statements

should not occur in "finally" blocks

Rug Bug

Rug Bug

R Bug

Allowing public ACLs or policies on a S3 bucket is security-sensitive

Security Hotspot

Using publicly writable directories is security-sensitive

Security Hotspot

Using clear-text protocols is security-sensitive

Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive

Security Hotspot

```
Add() + 1
1 + Add()
Add() + Empty()
Empty() + Add()
```

See

- Python documentation Rich comparison methods
- Python documentation Emulating numeric types

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