
































-  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... 🔍

The output of functions that don't return anything should not be used
 Bug
"=+" should not be used instead of "+="
 Bug
Increment and decrement operators should not be used
 Bug
Return values from functions without side effects should not be ignored
 Bug
Related "if/else if" statements should not have the same condition
 Bug
Identical expressions should not be used on both sides of a binary operator
 Bug
All code should be reachable
 Bug
Loops with at most one iteration should be refactored
 Bug
Variables should not be self-assigned
 Bug
All "except" blocks should be able to catch exceptions
 Bug
Constructing arguments of system commands from user input is security-sensitive
 Security Hotspot

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

Analyze your code

 Bug

 Critical



 regex

In regular expressions the boundaries `^` and `\A` can only match at the beginning of the input (or, in case of `^` in combination with the `MULTILINE` flag, the beginning of the line) and `$`, `\z` and `\Z` only at the end.

These patterns can be misused, by accidentally switching `^` and `$` for example, to create a pattern that can never match.

Noncompliant Code Example

```
# This can never match because $ and ^ have been switch
r"$[a-z]^+" # Noncompliant
```

Compliant Solution

```
r"^[a-z]+$"
```

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](#)


Disabling auto-escaping in template engines is security-sensitive

 Security Hotspot

Setting loose POSIX file permissions is security-sensitive

 Security Hotspot

Formatting SQL queries is security-sensitive

 Security Hotspot

Character classes in regular expressions should not contain only one character

 Code Smell