

- 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...



should be avoided

Code Smell

Non-capturing groups without quantifier should not be used

Code Smell

Regular expressions should not contain empty groups

Code Smell

Regular expressions should not contain multiple spaces

Code Smell

Single-character alternations in regular expressions should be replaced with character classes

Code Smell

Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string

Code Smell

Values assigned to variables should match their type annotations

Code Smell

Function return types should be consistent with their type hint

Code Smell

Character classes in regular expressions should not contain the same character twice

Code Smell

Type checks shouldn't be confusing

Code Smell

Regular expressions should not be too complicated

Code Smell

Signalling processes is security-sensitive

Analyze your code

Security Hotspot Critical ? cwe

Signalling processes is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2009-0390](#)
- [CVE-2002-0839](#)
- [CVE-2008-1671](#)

Sending signals without checking properly which process will receive it can cause a denial of service.

Ask Yourself Whether

- the PID of the process to which the signal will be sent is coming from an untrusted source. It could for example come from a world-writable file.
- users who are asking for the signal to be sent might not have the permission to send those signals.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- If the signal is sent because of a user's request. Check that the user is allowed to send this signal. You can for example forbid it if the user doesn't own the process.
- Secure the source from which the process PID is read.
- Run the process sending the signals with minimal permissions.

Sensitive Code Example

```
import os

def send_signal(pid, sig, pgid):
    os.kill(pid, sig) # Sensitive
    os.killpg(pgid, sig) # Sensitive
```

See

- [MITRE, CWE-283](#) - Unverified Ownership

Available In:

sonarcloud | sonarqube


Builtins should not be shadowed by local variables

 Code Smell

Implicit string and byte concatenations should not be confusing

 Code Smell

Identity comparisons should not be used with cached typed

 Code Smell

Expressions creating sets should not have duplicate values

 Code Smell