- 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...

methods should have the expected number of parameters

Bug

Special methods should have an expected number of parameters

Bug

Instance and class methods should have at least one positional parameter

Bug

Boolean expressions of exceptions should not be used in "except" statements

Bug

Caught Exceptions must derive from BaseException

Bug

Item operations should be done on objects supporting them

Bug

Raised Exceptions must derive from BaseException

Bug

Operators should be used on compatible types

Bug

Function arguments should be passed only once

Bug

Iterable unpacking, "for-in" loops and "yield from" should use an Iterable object

Bug

Variables, classes and functions should be defined before being used

Bug

XPath expressions should not be vulnerable to injection attacks

Analyze your code

Vulnerability Blocker injection cwe owasp

User-provided data, such as URL parameters, should always be considered untrusted and tainted. Constructing XPath expressions directly from tainted data enables attackers to inject specially crafted values that changes the initial meaning of the expression itself. Successful XPath injection attacks can read sensitive information from XML documents.

Noncompliant Code Example

Standard xml module (xml.etree.ElementTree) is not recommended:

- it provides a limited support of xpath expressions
- to parse untrusted xml for other security reasons
- does not have a way to parameterized xpath expressions

```
from flask import request
import xml.etree.ElementTree as ET

tree = ET.parse('users.xml')
root = tree.getroot()

@app.route('/user')
def user_location():
    username = request.args['username']
    query = "./users/user/[@name='"+username+"']/locati
    elmts = root.findall(query) # Noncompliant
    return 'Location %s' % list(elmts)
```

Compliant Solution

lxml module:

```
from flask import request
from lxml import etree

parser = etree.XMLParser(resolve_entities=False)
tree = etree.parse('users.xml', parser)
root = tree.getroot()

@app.route('/user')
def user_location():
    username = request.args['username']
    query = "/collection/users/user[@name = $paramname]
    elmts = root.xpath(query, paramname = username)
    return 'Location %s' % list(elmts)
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-643 - Improper Neutralization of Data within XPath Expressions

Available In:

sonarcloud sonarqube Developer Edition

Identity operators should not be used with dissimilar types

Bug

Only strings should be listed in "__all__"

Bug

"__init__" should not return a value

Bug

"yield" and "return" should not be used outside functions

Bug

String formatting should not lead to