sonar RULES

Products

- 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...*

Functions should not have too many lines of code
Code Smell

Track uses of "NOSONAR" comments
Code Smell

Track comments matching a regular expression
Code Smell

Statements should be on separate lines
Code Smell

Functions should not contain too many return statements
Code Smell

Files should not have too many lines of code
Code Smell

Lines should not be too long
Code Smell

Methods and properties that don't access instance data should be static
Code Smell

New-style classes should be used
Code Smell

Parentheses should not be used after certain keywords
Code Smell

Track "TODO" and "FIXME" comments that do not contain a reference to a person
Code Smell

Module names should comply with a naming convention

Logging should not be vulnerable to injection attacks

Analyze your code

Vulnerability | Minor | injection cwe owasp sans-top25

User-provided data, such as URL parameters, POST data payloads or cookies, should always be considered untrusted and tainted. Applications logging tainted data could enable an attacker to inject characters that would break the log file pattern. This could be used to block monitors and SIEM (Security Information and Event Management) systems from detecting other malicious events.

This problem could be mitigated by sanitizing the user-provided data before logging it.

Noncompliant Code Example

```
from flask import request, current_app
import logging

@app.route('/log')
def log():
    input = request.args.get('input')
    current_app.logger.error("%s", input) # Noncomplian
```

Compliant Solution

```
from flask import request, current_app
import logging

@app.route('/log')
def log():
    input = request.args.get('input')
    if input.isalnum():
        current_app.logger.error("%s", input) # Complia
```

See

- OWASP Top 10 2021 Category A9 - Security Logging and Monitoring Failures
- OWASP Cheat Sheet - Logging
- OWASP Attack Category - Log Injection
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-117 - Improper Output Neutralization for Logs
- SANS Top 25 - Insecure Interaction Between Components

Available In:
sonarcloud | sonarqube Developer Edition

Code Smell

Comments should not be located at the end of lines of code

Code Smell

Lines should not end with trailing whitespaces

Code Smell

Files should contain an empty newline at the end

Code Smell

Long suffix "L" should be upper case

Code Smell