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...

Code Smell

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

Constructing arguments of system commands from user input is security-sensitive

Analyze your code

Security Hotspot | Major | injection cwe owasp sans-top25

Constructing arguments of system commands from user input is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2016-9920
- CVE-2021-29472

Arguments of system commands are processed by the executed program. The arguments are usually used to configure and influence the behavior of the programs. Control over a single argument might be enough for an attacker to trigger dangerous features like executing arbitrary commands or writing files into specific directories.

Ask Yourself Whether

- Malicious arguments can result in undesired behavior in the executed command.
- Passing user input to a system command is not necessary.

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

Recommended Secure Coding Practices

- Avoid constructing system commands from user input when possible.
- Ensure that no risky arguments can be injected for the given program, e.g., type-cast the argument to an integer.
- Use a more secure interface to communicate with other programs, e.g., the standard input stream (stdin).

Sensitive Code Example

Arguments like `-delete` or `-exec` for the `find` command can alter the expected behavior and result in vulnerabilities:

```
import subprocess
input = request.get('input')
subprocess.run(["/usr/bin/find", input]) # Sensitive
```

Compliant Solution

Use an allow-list to restrict the arguments to trusted values:

```
import subprocess
input = request.get('input')
if input in allowed:
    subprocess.run(["/usr/bin/find", input])
```

Code Smell

Module names should comply with a naming convention

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

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-88 - Argument Injection or Modification
- SANS Top 25 - Insecure Interaction Between Components
- CVE-2021-29472 - PHP Supply Chain Attack on Composer

Available In:

sonarcloud | sonarqube Developer Edition