

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- Code Smell
- 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
Code Smell

Setting loose POSIX file permissions is security-sensitive

Analyze your code

Security Hotspot Major cwe sans-top25 owasp

In Unix, "others" class refers to all users except the owner of the file and the members of the group assigned to this file.

Granting permissions to this group can lead to unintended access to files.

Ask Yourself Whether

- The application is designed to be run on a multi-user environment.
- Corresponding files and directories may contain confidential information.

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

Recommended Secure Coding Practices

The most restrictive possible permissions should be assigned to files and directories.

Sensitive Code Example

For `os.umask`:

```
os.umask(0) # Sensitive
```

For `os.chmod`, `os.lchmod`, and `os.fchmod`:

```
os.chmod("/tmp/fs", stat.S_IRWXO) # Sensitive
os.lchmod("/tmp/fs", stat.S_IRWXO) # Sensitive
os.fchmod(fd, stat.S_IRWXO) # Sensitive
```

Compliant Solution

For `os.umask`:

```
os.umask(0o777)
```


For `os.chmod`, `os.lchmod`, and `os.fchmod`:

```
os.chmod("/tmp/fs", stat.S_IRWXU)
os.lchmod("/tmp/fs", stat.S_IRWXU)
os.fchmod(fd, stat.S_IRWXU)
```


See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2021 Category A4 - Insecure Design


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

Long suffix "L" should be upper case

- [OWASP Top 10 2017 Category A5](#) - Broken Access Control
- [OWASP File Permission](#)
- [MITRE, CWE-732](#) - Incorrect Permission Assignment for Critical Resource
- [MITRE, CWE-266](#) - Incorrect Privilege Assignment
- [SANS Top 25](#) - Porous Defenses

Available In:

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](#)