

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- 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...

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

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

Using weak hashing algorithms is security-sensitive

Analyze your code

Security Hotspot

Critical

cwe spring owasp sans-top25

Cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, HMAC-MD5, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160, HMACRIPEMD160 and SHA-1 are no longer considered secure, because it is possible to have collisions (little computational effort is enough to find two or more different inputs that produce the same hash).

Ask Yourself Whether

The hashed value is used in a security context like:

- User-password storage.
- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...).
- To compute some message integrity.

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

Recommended Secure Coding Practices

Safer alternatives, such as SHA-256, SHA-512, SHA-3 are recommended, and for password hashing, it's even better to use algorithms that do not compute too "quickly", like bcrypt, scrypt, argon2 or pbkdf2 because it slows down brute force attacks.

Sensitive Code Example

```
import hashlib
m = hashlib.md5() // Sensitive
```

```
import hashlib
m = hashlib.sha1() // Sensitive
```


```
import md5 // Sensitive and deprecated since Python 2.5
m = md5.new()

import sha // Sensitive and deprecated since Python 2.5
m = sha.new()
```


Compliant Solution

```
import hashlib
m = hashlib.sha512() // Compliant
```


used with cached typed

 Code Smell


Expressions creating sets should not have duplicate values

 Code Smell

Expressions creating dictionaries should not have duplicate keys

 Code Smell

Special method "__exit__" should not re-raise the provided exception

 Code Smell

Unused scope-limited definitions should be removed

See

- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [Mobile AppSec Verification Standard](#) - Cryptography Requirements
- [OWASP Mobile Top 10 2016 Category M5](#) - Insufficient Cryptography
- [MITRE, CWE-1240](#) - Use of a Risky Cryptographic Primitive
- [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](#)