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...

generators (PRNGs) is security-sensitive
Security Hotspot

Constants should not be used as conditions
Code Smell

"SystemExit" should be re-raised
Code Smell

Bare "raise" statements should only be used in "except" blocks
Code Smell

Comparison to None should not be constant
Code Smell

"self" should be the first argument to instance methods
Code Smell

Function parameters' default values should not be modified or assigned
Code Smell

Some special methods should return "NotImplemented" instead of raising "NotImplementedError"
Code Smell

Custom Exception classes should inherit from "Exception" or one of its subclasses
Code Smell

Bare "raise" statements should not be used in "finally" blocks
Code Smell

Arguments given to functions should be of an expected type
Code Smell

Hard-coded credentials are security-sensitive

Analyze your code

Security Hotspot | Blocker | cwe sans-top25 owasp

Because it is easy to extract strings from an application source code or binary, credentials should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- CVE-2019-13466
- CVE-2018-15389

Credentials should be stored outside of the code in a configuration file, a database, or a management service for secrets.

This rule flags instances of hard-coded credentials used in database and LDAP connections. It looks for hard-coded credentials in connection strings, and for variable names that match any of the patterns from the provided list.

It's recommended to customize the configuration of this rule with additional credential words such as "oauthToken", "secret", ...

Ask Yourself Whether

- Credentials allow access to a sensitive component like a database, a file storage, an API or a service.
- Credentials are used in production environments.
- Application re-distribution is required before updating the credentials.

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

Recommended Secure Coding Practices

- Store the credentials in a configuration file that is not pushed to the code repository.
- Store the credentials in a database.
- Use your cloud provider's service for managing secrets.
- If a password has been disclosed through the source code: change it.

Sensitive Code Example

```
username = 'admin'
password = 'admin' # Sensitive
usernamePassword = 'user=admin&password=admin' # Sensit
```

Compliant Solution

```
import os

username = os.getenv("username") # Compliant
password = os.getenv("password") # Compliant
usernamePassword = 'user=%s&password=%s' % (username, p
```

`str.replace` should be preferred to `re.sub`

Code Smell

Unread "private" attributes should be removed

Code Smell

Cognitive Complexity of functions should not be too high

Code Smell

The first argument to class methods should follow the naming convention

Code Smell

Method overrides should not change contracts

See

- OWASP Top 10 2021 Category A7 - Identification and Authentication Failures
- OWASP Top 10 2017 Category A2 - Broken Authentication
- MITRE, CWE-798 - Use of Hard-coded Credentials
- MITRE, CWE-259 - Use of Hard-coded Password
- SANS Top 25 - Porous Defenses
- Derived from FindSecBugs rule Hard Coded Password

Available In:

sonarcloud | sonarqube