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

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

Analyze your code

Security Hotspot   Minor   cwe owasp sans-top25

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2018-0269
- CVE-2017-14460

Same origin policy in browsers prevents, by default and for security-reasons, a javascript frontend to perform a cross-origin HTTP request to a resource that has a different origin (domain, protocol, or port) from its own. The requested target can append additional HTTP headers in response, called CORS, that act like directives for the browser and change the access control policy / relax the same origin policy.

Ask Yourself Whether

- You don't trust the origin specified, example: `Access-Control-Allow-Origin: untrustedwebsite.com`.
- Access control policy is entirely disabled: `Access-Control-Allow-Origin: *`
- Your access control policy is dynamically defined by a user-controlled input like `origin` header.

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

Recommended Secure Coding Practices

- The `Access-Control-Allow-Origin` header should be set only for a trusted origin and for specific resources.
- Allow only selected, trusted domains in the `Access-Control-Allow-Origin` header. Prefer whitelisting domains over blacklisting or allowing any domain (do not use * wildcard nor blindly return the `Origin` header content without any checks).

Sensitive Code Example

Django:

```
CORS_ORIGIN_ALLOW_ALL = True # Sensitive
```

Flask:

```
from flask import Flask
from flask_cors import CORS

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "*", "send_wild
```

User-controlled origin:

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

```
origin = request.headers['ORIGIN']
resp = Response()
resp.headers['Access-Control-Allow-Origin'] = origin #
```

Compliant Solution

Django:

```
CORS_ORIGIN_ALLOW_ALL = False # Compliant
```

Flask:

```
from flask import Flask
from flask_cors import CORS

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "*", "send_wild
```

User-controlled origin validated with an allow-list:

```
origin = request.headers['ORIGIN']
resp = Response()
if origin in TRUSTED_ORIGINS:
   resp.headers['Access-Control-Allow-Origin'] = origin
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2021 Category A7 - Identification and Authentication Failures
- developer.mozilla.org - CORS
- developer.mozilla.org - Same origin policy
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- OWASP HTML5 Security Cheat Sheet - Cross Origin Resource Sharing
- MITRE, CWE-346 - Origin Validation Error
- MITRE, CWE-942 - Overly Permissive Cross-domain Whitelist
- SANS Top 25 - Porous Defenses

Available In:

sonarcloud sonarqube