



Text 月

тѕ 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) 6 Vulnerability (29)

R Bug (55)

Security Hotspot 31

Code Smell (101)

runctions snould not have too many lines of code A 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 A Code Smell

Track "TODO" and "FIXME" comments

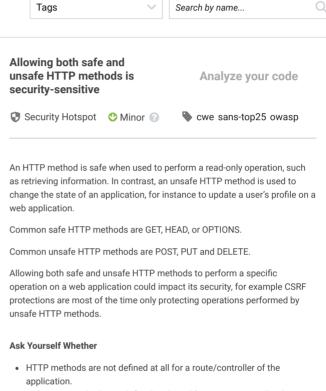
that do not contain a reference to a

Module names should comply with a

person

Code Smell

naming convention



Safe HTTP methods are defined and used for a route/controller that can change the state of an application.

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

Recommended Secure Coding Practices

For all the routes/controllers of an application, the authorized HTTP methods should be explicitly defined and safe HTTP methods should only be used to perform read-only operations.

Sensitive Code Example

For Django:

```
# No method restriction
def view(request): # Sensitive
    return HttpResponse("...")
```

```
@require http methods(["GET", "POST"]) # Sensitive
def view(request):
   return HttpResponse("...")
```

For Flask:

```
@methods.route('/sensitive', methods=['GET', 'POST'])
def view():
   return Response("...", 200)
```

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

Compliant Solution

For Django:

```
@require_http_methods(["POST"])
def view(request):
    return HttpResponse("...")
```

```
@require_POST
def view(request):
    return HttpResponse("...")
```

```
@require_GET
def view(request):
    return HttpResponse("...")
```

```
@require_safe
def view(request):
    return HttpResponse("...")
```

For Flask:

```
@methods.route('/compliant1')
def view():
    return Response("...", 200)
```

```
@methods.route('/compliant2', methods=['GET'])
def view():
    return Response("...", 200)
```

See

- OWASP Top 10 2021 Category A1 Broken Access Control
- OWASP Top 10 2021 Category A4 Insecure Design
- OWASP Top 10 2017 Category A5 Broken Access Control
- MITRE, CWE-352 Cross-Site Request Forgery (CSRF)
- OWASP: Cross-Site Request Forgery
- SANS Top 25 Insecure Interaction Between Components
- Django Allowed HTTP Methods
- Flask HTTP Methods

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