

- 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

Allowing both safe and unsafe HTTP methods is security-sensitive

Analyze your code

Security Hotspot Minor cwe sans-top25 owasp

An HTTP method is safe when used to perform a read-only operation, such as retrieving information. In contrast, an unsafe HTTP method is used to change the state of an application, for instance to update a user's profile on a web application.

Common safe HTTP methods are GET, HEAD, or OPTIONS.

Common unsafe HTTP methods are POST, PUT and DELETE.

Allowing both safe and unsafe HTTP methods to perform a specific operation on a web application could impact its security, for example CSRF protections are most of the time only protecting operations performed by unsafe HTTP methods.

Ask Yourself Whether

- HTTP methods are not defined at all for a route/controller of the application.
- 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:

sonardcloud  | **sonarqube** 