

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- Code Smell
- 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

Creating cookies without the "HttpOnly" flag is security-sensitive

Analyze your code

Security Hotspot

Minor ?

cwe sans-top25 privacy owasp

When a cookie is configured with the `HttpOnly` attribute set to `true`, the browser guaranties that no client-side script will be able to read it. In most cases, when a cookie is created, the default value of `HttpOnly` is `false` and it's up to the developer to decide whether or not the content of the cookie can be read by the client-side script. As a majority of Cross-Site Scripting (XSS) attacks target the theft of session-cookies, the `HttpOnly` attribute can help to reduce their impact as it won't be possible to exploit the XSS vulnerability to steal session-cookies.

Ask Yourself Whether

- the cookie is sensitive, used to authenticate the user, for instance a *session-cookie*
- the `HttpOnly` attribute offer an additional protection (not the case for an *XSRF-TOKEN cookie* / CSRF token for example)

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

Recommended Secure Coding Practices

- By default the `HttpOnly` flag should be set to `true` for most of the cookies and it's mandatory for session / sensitive-security cookies.

Sensitive Code Example

Flask:

```
from flask import Response


@app.route('/')
def index():
    response = Response()
    response.set_cookie('key', 'value') # Sensitive
    return response
```

Compliant Solution


Flask:

```
from flask import Response


@app.route('/')
def index():
    response = Response()
    response.set_cookie('key', 'value', httponly=True)
    return response
```

 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

See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [OWASP HttpOnly](#)
- [OWASP Top 10 2017 Category A7](#) - Cross-Site Scripting (XSS)
- [MITRE, CWE-1004](#) - Sensitive Cookie Without 'HttpOnly' Flag
- [SANS Top 25](#) - Insecure Interaction Between Components
- Derived from FindSecBugs rule [HTTPONLY_COOKIE](#)

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](#)