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...

HTTP responses should not be vulnerable to session fixation
Vulnerability

Dynamic code execution should not be vulnerable to injection attacks
Vulnerability

NoSQL operations should not be vulnerable to injection attacks
Vulnerability

HTTP request redirections should not be open to forging attacks
Vulnerability

Deserialization should not be vulnerable to injection attacks
Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks
Vulnerability

Database queries should not be vulnerable to injection attacks
Vulnerability

XML parsers should not be vulnerable to XXE attacks
Vulnerability

A secure password should be used when connecting to a database
Vulnerability

XPath expressions should not be vulnerable to injection attacks
Vulnerability

I/O function calls should not be vulnerable to path injection attacks
Vulnerability

HTTP responses should not be vulnerable to session fixation

Analyze your code

Vulnerability | Blocker | injection cwe owasp

User-provided data, such as URL parameters, should always be considered untrusted and tainted. Constructing cookies directly from tainted data enables attackers to set the session identifier to a known value, allowing the attacker to share the session with the victim. Successful attacks might result in unauthorized access to sensitive information, for example if the session identifier is not regenerated when the victim authenticates.

Typically, the solution to prevent this type of attack is to restrict the cookies that can be influenced with an allow-list.

Noncompliant Code Example

```
from django.http import HttpResponse

def index(request):
    value = request.GET.get("value")
    response = HttpResponse("")
    response["Set-Cookie"] = value  # Noncompliant
    response.set_cookie("sessionid", value)  # Noncompl
    return response
```

Compliant Solution

```
from django.http import HttpResponse

def index(request):
    value = request.GET.get("value")
    response = HttpResponse("")
    response["X-Data"] = value
    response.set_cookie("data", value)
    return response
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-384 - Session Fixation

Available In:

sonarcloud | sonarqube Developer Edition

LDAP queries should not be vulnerable to injection attacks

Vulnerability

OS commands should not be vulnerable to command injection attacks

Vulnerability

The number and name of arguments passed to a function should match its parameters

Bug

The "open" builtin function should be called with a valid mode

Bug