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...*

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

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

HTTP request redirections should not be open to forging attacks

Analyze your code

Vulnerability | Blocker | injection cwe sans-top25 owasp

User-provided data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Applications performing HTTP redirects based on tainted data could enable an attacker to redirect users to a malicious site to, for example, steal login credentials.

This problem could be mitigated in any of the following ways:

- Validate the user-provided data based on an allowlist and reject input not matching.
- Redesign the application to not perform redirects based on user-provided data.

Noncompliant Code Example

Flask

```
from flask import request, redirect, Response

@app.route('flask_redirect')
def flask_redirect():
    url = request.args["next"]
    return redirect(url)  # Noncompliant

@app.route('set_location_header')
def set_location_header():
    url = request.args["next"]
    response = Response("redirecting...", 302)
    response.headers['Location'] = url  # Noncompliant
    return response
```

Django

```
from django.http import HttpResponseRedirect

def http_responser_redirect(request):
    url = request.GET.get("next", "/")
    return HttpResponseRedirect(url)  # Noncompliant

def set_location_header(request):
    url = request.GET.get("next", "/")
    response = HttpResponse(status=302)
    response['Location'] = url  # Noncompliant
    return response
```

Compliant Solution

Only defined names should be listed in "__all__"

Bug

Calls should not be made to non-callable values

Bug

Property getter, setter and deleter methods should have the expected number of parameters

Bug

Special methods should have an expected number of parameters

Bug

Instance and class methods should

Flask

```
from flask import request, redirect, Response, url_for

@app.route('flask_redirect')
def flask_redirect():
    endpoint = request.args["next"]
    return redirect(url_for(endpoint))  # Compliant
```

Django

```
from django.http import HttpResponseRedirect
from urllib.parse import urlparse

DOMAINS_WHITELIST = ['www.example.com', 'example.com']

def http_responser_redirect(request):
    url = request.GET.get("next", "/")
    parsed_uri = urlparse(url)
    if parsed_uri.netloc in DOMAINS_WHITELIST:
        return HttpResponseRedirect(url)  # Compliant
    return HttpResponseRedirect("/")
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-601 - URL Redirection to Untrusted Site ('Open Redirect')
- SANS Top 25 - Risky Resource Management

Available In:

sonarcloud sonarqube Developer Edition