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...

should not be used in "except" statements
Bug

Caught Exceptions must derive from BaseException
Bug

Item operations should be done on objects supporting them
Bug

Raised Exceptions must derive from BaseException
Bug

Operators should be used on compatible types
Bug

Function arguments should be passed only once
Bug

Iterable unpacking, "for-in" loops and "yield from" should use an Iterable object
Bug

Variables, classes and functions should be defined before being used
Bug

Identity operators should not be used with dissimilar types
Bug

Only strings should be listed in "__all__"
Bug

"__init__" should not return a value
Bug

"yield" and "return" should not be used

OS commands should not be vulnerable to command injection attacks

Analyze your code

Vulnerability | Blocker | injection cwe owasp sans-top25

Applications that execute operating system commands or execute commands that interact with the underlying system should neutralize any externally-provided values used in those commands. Failure to do so could allow an attacker to include input that executes unintended commands or exposes sensitive data.

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

- Using subprocess module without the `shell=true`. In this case `subprocess` expects an array where command and arguments are clearly separated.
- Escaping shell argument with shlex.quote

Noncompliant Code Example

os

```
from flask import request
import os

@app.route('/ping')
def ping():
    address = request.args.get("address")
    cmd = "ping -c 1 %s" % address
    os.popen(cmd) # Noncompliant
```

subprocess

```
from flask import request
import subprocess

@app.route('/ping')
def ping():
    address = request.args.get("address")
    cmd = "ping -c 1 %s" % address
    subprocess.Popen(cmd, shell=True) # Noncompliant; u
```

Compliant Solution

os

```
from flask import request
import os

@app.route('/ping')
def ping():
```

'yield' and 'return' should not be used outside functions

Bug

String formatting should not lead to runtime errors

Bug

Recursion should not be infinite

Bug

Silly equality checks should not be made

Bug

Granting access to S3 buckets to all or authenticated users is security-sensitive

```
    address = shlex.quote(request.args.get("address"))
    cmd = "ping -c 1 %s" % address
    os.popen(cmd ) # Compliant
```

`subprocess`

```
from flask import request
import subprocess

@app.route('/ping')
def ping():
    address = request.args.get("address")
    args = ["ping", "-c1", address]
    subprocess.Popen(args) # Compliant
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP OS Command Injection Defense Cheat Sheet
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-78 - Improper Neutralization of Special Elements used in an OS Command
- SANS Top 25 - Insecure Interaction Between Components