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...

Code Smell

`str.replace` should be preferred to `re.sub`
Code Smell

Unread "private" attributes should be removed
Code Smell

Cognitive Complexity of functions should not be too high
Code Smell

The first argument to class methods should follow the naming convention
Code Smell

Method overrides should not change contracts
Code Smell

Wildcard imports should not be used
Code Smell

String literals should not be duplicated
Code Smell

Functions and methods should not be empty
Code Smell

Server-side requests should not be vulnerable to forging attacks
Vulnerability

Non-empty statements should change control flow or have at least one side-effect
Bug

Replacement strings should reference existing regular expression groups
Bug

Alternation in regular expressions should not contain empty alternatives

Cipher algorithms should be robust

Analyze your code

Vulnerability | Critical | cwe privacy owasp sans-top25

Strong cipher algorithms are cryptographic systems resistant to cryptanalysis, they are not vulnerable to well-known attacks like brute force attacks for example.

A general recommendation is to only use cipher algorithms intensively tested and promoted by the cryptographic community.

More specifically for block cipher, it's not recommended to use algorithm with a block size inferior than 128 bits.

Noncompliant Code Example

pycryptodomex library:

```
from Cryptodome.Cipher import DES, DES3, ARC2, ARC4, Blowfis
from Cryptodome.Random import get_random_bytes

key = b'-8B key-'
DES.new(key, DES.MODE_OFB) # Noncompliant: DES works with 56

key = DES3.adjust_key_parity(get_random_bytes(24))
cipher = DES3.new(key, DES3.MODE_CFB) # Noncompliant: Triple

key = b'Sixteen byte key'
cipher = ARC2.new(key, ARC2.MODE_CFB) # Noncompliant: RC2 is

key = b'Very long and confidential key'
cipher = ARC4.new(key) # Noncompliant: vulnerable to several

key = b'An arbitrarily long key'
cipher = Blowfish.new(key, Blowfish.MODE_CBC) # Noncompliant
```

pycryptodome library:

```
from Crypto.Cipher import DES, DES3, ARC2, ARC4, Blowfish, A
from Crypto.Random import get_random_bytes

key = b'-8B key-'
DES.new(key, DES.MODE_OFB) # Noncompliant: DES works with 56

key = DES3.adjust_key_parity(get_random_bytes(24))
cipher = DES3.new(key, DES3.MODE_CFB) # Noncompliant: Triple

key = b'Sixteen byte key'
cipher = ARC2.new(key, ARC2.MODE_CFB) # Noncompliant: RC2 is

key = b'Very long and confidential key'
cipher = ARC4.new(key) # Noncompliant: vulnerable to several

key = b'An arbitrarily long key'
cipher = Blowfish.new(key, Blowfish.MODE_CBC) # Noncompliant
```

Bug

Unicode Grapheme Clusters should be avoided inside regex character classes

Bug

Regex alternatives should not be redundant

Bug

Alternatives in regular expressions should be grouped when used with anchors

Bug

pyca library:

```
import os
from cryptography.hazmat.primitives.ciphers import Cipher, a
from cryptography.hazmat.backends import default_backend

key = os.urandom(16)
iv = os.urandom(16)

tdes4 = Cipher(algorithms.TripleDES(key), mode=None, backend
bf3 = Cipher(algorithms.Blowfish(key), mode=None, backend=de
rc42 = Cipher(algorithms.ARC4(key), mode=None, backend=defau
```

pydes library:

```
import pyDes;

des1 = pyDes.des('ChangeIt')  # Noncompliant: DES works with
des2 = pyDes.des('ChangeIt', pyDes.CBC, "\0\0\0\0\0\0\0\0",

tdes1 = pyDes.triple_des('ChangeItWithYourKey!!!!!')  # Nonc
tdes2 = pyDes.triple_des('ChangeItWithYourKey!!!!!', pyDes.C
```

pycrypto library is not maintained and therefore should not be used:

```
from Crypto.Cipher import *

des3 = DES.new('ChangeIt') # Noncompliant: DES works with 56
tdes3 = DES3.new('ChangeItChangeIt') # Noncompliant: Triple
bf2 = Blowfish.new('ChangeItWithYourKey', Blowfish.MODE_CBC,
rc21 = ARC2.new('ChangeItWithYourKey', ARC2.MODE_CFB, 'Chang
rc41 = ARC4.new('ChangeItWithYourKey') # Noncompliant: vulne
```

Compliant Solution

pycryptodomex library:

```
from Cryptodome.Cipher import AES

key = b'Sixteen byte key'
cipher = AES.new(key, AES.MODE_CCM) # Compliant
```

pycryptodome library:

```
from Crypto.Cipher import AES

key = b'Sixteen byte key'
cipher = AES.new(key, AES.MODE_CCM) # Compliant
```

pyca library:

```
import os
from cryptography.hazmat.primitives.ciphers import Cipher, a
from cryptography.hazmat.backends import default_backend

key = os.urandom(16)
iv = os.urandom(16)

aes2 = Cipher(algorithms.AES(key), modes.CBC(iv), backend=de
```

pycrypto library is not maintained and therefore should not be used:

```
from Crypto.Cipher import *

aes1 = AES.new('This is a key123', AES.MODE_CBC, 'This is an
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-327 - Use of a Broken or Risky Cryptographic Algorithm
- SANS Top 25 - Porous Defenses

Available In:
sonarlint | sonarcloud | sonarqube