Search by name...





Java

Kotlin

PHP

PL/I

PL/SQL

**Python** 

**RPG** 

Ruby

Scala

Swift

Text

T-SQL

**VB.NET** 

VB<sub>6</sub>

**XML** 

Terraform

**TypeScript** 

Oil

1

JavaScript

Objective C



# Python static code analysis

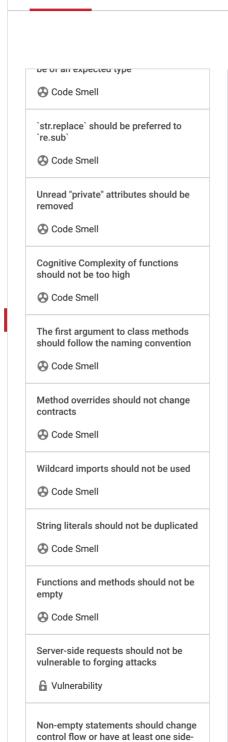
Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PYTHON code

All rules 216	6 Vulnerability (29)	<b>∰</b> Bug 55	Security Hotspot 31	Code Smell (101)

pyca library:

import os

Tags

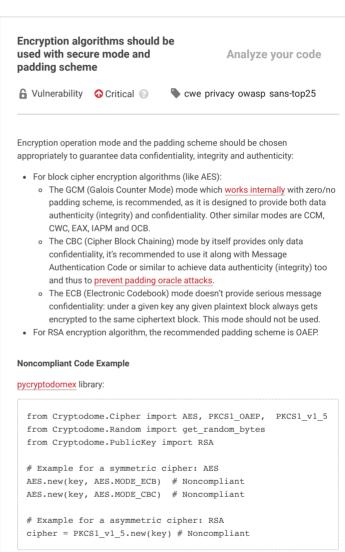


effect

₩ Bug

₩ Bug

Replacement strings should reference existing regular expression groups



from cryptography.hazmat.primitives.ciphers import Cipher, a from cryptography.hazmat.backends import default\_backend

from cryptography.hazmat.primitives.asymmetric import rsa, p

aes = Cipher(algorithms.AES(key), modes.CBC(iv), backend=def

aes = Cipher(algorithms.AES(key), modes.ECB(), backend=defau

from cryptography.hazmat.primitives import hashes

# Example for a symmetric cipher: AES

# Example for a asymmetric cipher: RSA ciphertext = public\_key.encrypt(

padding.PKCS1v15() # Noncompliant

plaintext = private\_key.decrypt(

Alternation in regular expressions should not contain empty alternatives

# Bua

Unicode Grapheme Clusters should be avoided inside regex character classes

₩ Bua

Regex alternatives should not be redundant

👬 Bug

Alternatives in regular expressions should be grouped when used with anchors

```
ciphertext,
padding.PKCS1v15() # Noncompliant
)
```

## pydes library:

```
# For DES cipher
des = pyDes.des('ChangeIt') # Noncompliant
des = pyDes.des('ChangeIt', pyDes.CBC, "\0\0\0\0\0\0\0\0\0", p
des = pyDes.des('ChangeIt', pyDes.ECB, "\0\0\0\0\0\0\0\0\0\0", p
```

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

```
# https://pycrypto.readthedocs.io/en/latest/
from Crypto.Cipher import *
from Crypto.Random import get_random_bytes
from Crypto.Util import Counter
from Crypto.PublicKey import RSA

# Example for a symmetric cipher: AES
AES.new(key, AES.MODE_ECB) # Noncompliant
AES.new(key, AES.MODE_CBC, IV=iv) # Noncompliant
# Example for a asymmetric cipher: RSA
cipher = PKCS1_v1_5.new(key) # Noncompliant
```

#### Compliant Solution

pycryptodomex library:

```
from Cryptodome.Cipher import AES
from Cryptodome.Random import get_random_bytes
from Cryptodome.PublicKey import RSA

# AES is the recommended symmetric cipher with GCM mode
AES.new(key, AES.MODE_GCM) # Compliant

# RSA is the recommended asymmetric cipher with OAEP padding
cipher = PKCS1_OAEP.new(key) # Compliant
```

## pyca library:

```
import os
from cryptography.hazmat.primitives.ciphers import Cipher, a
from cryptography.hazmat.backends import default backend
from cryptography.hazmat.primitives.asymmetric import rsa, p
from cryptography.hazmat.primitives import hashes
# AES is the recommended symmetric cipher with GCM mode
aes = Cipher(algorithms.AES(key), modes.GCM(iv), backend=def
# RSA is the recommended asymmetric cipher with OAEP padding
ciphertext = public_key.encrypt(
 message,
  padding.OAEP( # Compliant
    mgf=padding.MGF1(algorithm=hashes.SHA256()),
    algorithm=hashes.SHA256(),
    label=None
plaintext = private_key.decrypt(
 ciphertext,
 padding.OAEP( # Compliant
    mgf=padding.MGF1(algorithm=hashes.SHA256()),
    algorithm=hashes.SHA256(),
    label=None
```

### See

- OWASP Top 10 2021 Category A2 Cryptographic Failures
- OWASP Top 10 2017 Category A6 Security Misconfiguration
- MITRE, CWE-327 Use of a Broken or Risky Cryptographic Algorithm
- SANS Top 25 Porous Defenses

Available In:
sonarlint ⊖ | 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