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...

Alternation in regular expressions should not contain empty alternatives
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

New objects should not be created only to check their identity
Bug

Collection content should not be replaced unconditionally
Bug

Exceptions should not be created without being raised
Bug

Collection sizes and array length comparisons should make sense
Bug

All branches in a conditional structure should not have exactly the same implementation
Bug

The output of functions that don't return anything should not be used
Bug

"=+" should not be used instead of "+="

Cipher Block Chaining IVs should be unpredictable

Analyze your code

Vulnerability | Critical | cwe owasp

When encrypting data with the Cipher Block Chaining (CBC) mode an Initialization Vector (IV) is used to randomize the encryption, ie under a given key the same plaintext doesn't always produce the same ciphertext. The IV doesn't need to be secret but should be unpredictable to avoid "Chosen-Plaintext Attack".

To generate Initialization Vectors, NIST recommends to use a secure random number generator.

Noncompliant Code Example

For PyCryptodome module:

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad

static_vector = b'x' * AES.block_size
cipher = AES.new(key, AES.MODE_CBC, static_vector)
cipher.encrypt(pad(data, AES.block_size))  # Noncomplia
```

For cryptography module:

```
from os import urandom
from cryptography.hazmat.primitives.ciphers import Ciph

static_vector = b'x' * 16
cipher = Cipher(algorithms.AES(key), modes.CBC(static_v
cipher.encryptor()  # Noncompliant
```

Compliant Solution

For PyCryptodome module:

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad

random_vector = get_random_bytes(AES.block_size)
cipher = AES.new(key, AES.MODE_CBC, random_vector)
cipher.encrypt(pad(data, AES.block_size))
```

For cryptography module:

```
from os import urandom
from cryptography.hazmat.primitives.ciphers import Ciph
```

Bug

Increment and decrement operators should not be used

Bug

Return values from functions without side effects should not be ignored

Bug

Related "if/else if" statements should not have the same condition

Bug

Identical expressions should not be used on both sides of a binary operator

```
random_vector = urandom(16)
cipher = Cipher(algorithms.AES(key), modes.CBC(random_v
cipher.encryptor()
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Cryptography Requirements
- OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography
- MITRE, CWE-329 - Not Using an Unpredictable IV with CBC Mode
- MITRE, CWE-330 - Use of Insufficiently Random Values
- MITRE, CWE-340 - Generation of Predictable Numbers or Identifiers
- MITRE, CWE-1204 - Generation of Weak Initialization Vector (IV)
- NIST, SP-800-38A - Recommendation for Block Cipher Modes of Operation

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