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...

Setting loose POSIX file permissions is security-sensitive
Security Hotspot

Formatting SQL queries is security-sensitive
Security Hotspot

Character classes in regular expressions should not contain only one character
Code Smell

Superfluous curly brace quantifiers should be avoided
Code Smell

Non-capturing groups without quantifier should not be used
Code Smell

Regular expressions should not contain empty groups
Code Smell

Regular expressions should not contain multiple spaces
Code Smell

Single-character alternations in regular expressions should be replaced with character classes
Code Smell

Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string
Code Smell

Values assigned to variables should match their type annotations
Code Smell

Function return types should be consistent with their type hint
Code Smell

Expanding archive files without controlling resource consumption is security-sensitive

Analyze your code

Security Hotspot | Critical | cwe owasp

Successful Zip Bomb attacks occur when an application expands untrusted archive files without controlling the size of the expanded data, which can lead to denial of service. A Zip bomb is usually a malicious archive file of a few kilobytes of compressed data but turned into gigabytes of uncompressed data. To achieve this extreme compression ratio, attackers will compress irrelevant data (eg: a long string of repeated bytes).

Ask Yourself Whether

Archives to expand are untrusted and:

- There is no validation of the number of entries in the archive.
- There is no validation of the total size of the uncompressed data.
- There is no validation of the ratio between the compressed and uncompressed archive entry.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Define and control the ratio between compressed and uncompressed data, in general the data compression ratio for most of the legit archives is 1 to 3.
- Define and control the threshold for maximum total size of the uncompressed data.
- Count the number of file entries extracted from the archive and abort the extraction if their number is greater than a predefined threshold, in particular it's not recommended to recursively expand archives (an entry of an archive could be also an archive).

Sensitive Code Example

For tarfile module:

```
import tarfile

tfile = tarfile.open("TarBomb.tar")
tfile.extractall('./tmp/')  # Sensitive
tfile.close()
```

For zipfile module:

```
import zipfile

zfile = zipfile.ZipFile('ZipBomb.zip', 'r')
zfile.extractall('./tmp/') # Sensitive
zfile.close()
```

Compliant Solution

For tarfile module:

Character classes in regular expressions should not contain the same character twice

Code Smell

Type checks shouldn't be confusing

Code Smell

Regular expressions should not be too complicated

Code Smell

Builtins should not be shadowed by local variables

Code Smell

```
import tarfile

THRESHOLD_ENTRIES = 10000
THRESHOLD_SIZE = 1000000000
THRESHOLD_RATIO = 10

totalSizeArchive = 0;
totalEntryArchive = 0;

tfile = tarfile.open("TarBomb.tar")
for entry in tfile:
  tarinfo = tfile.extractfile(entry)

  totalEntryArchive += 1
  sizeEntry = 0
  result = b''
  while True:
    sizeEntry += 1024
    totalSizeArchive += 1024

    ratio = sizeEntry / entry.size
    if ratio > THRESHOLD_RATIO:
      # ratio between compressed and uncompressed data is hi
      break

    chunk = tarinfo.read(1024)
    if not chunk:
      break

    result += chunk

  if totalEntryArchive > THRESHOLD_ENTRIES:
    # too much entries in this archive, can lead to inodes e
    break

  if totalSizeArchive > THRESHOLD_SIZE:
    # the uncompressed data size is too much for the applica
    break

tfile.close()
```

For zipfile module:

```
import zipfile

THRESHOLD_ENTRIES = 10000
THRESHOLD_SIZE = 1000000000
THRESHOLD_RATIO = 10

totalSizeArchive = 0;
totalEntryArchive = 0;

zfile = zipfile.ZipFile('ZipBomb.zip', 'r')
for zinfo in zfile.infolist():
    print('File', zinfo.filename)
    data = zfile.read(zinfo)

    totalEntryArchive += 1

    totalSizeArchive = totalSizeArchive + len(data)
    ratio = len(data) / zinfo.compress_size
    if ratio > THRESHOLD_RATIO:
      # ratio between compressed and uncompressed data is hi
      break

    if totalSizeArchive > THRESHOLD_SIZE:
      # the uncompressed data size is too much for the appli
      break

    if totalEntryArchive > THRESHOLD_ENTRIES:
      # too much entries in this archive, can lead to inodes
      break

zfile.close()
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- MITRE, CWE-409 - Improper Handling of Highly Compressed Data (Data Amplification)
- bamsoftware.com - A better Zip Bomb

Available In:

sonarcloud | sonarqube