



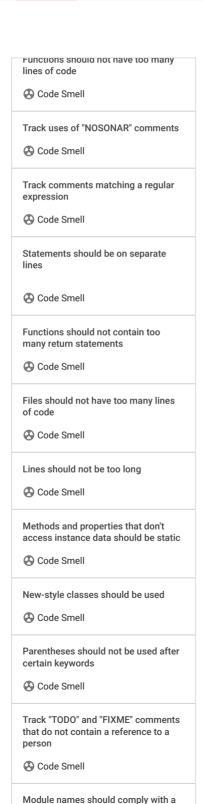


# Python static code analysis

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

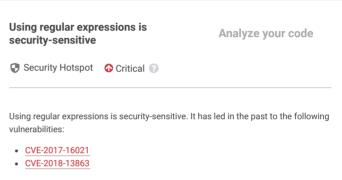
All rules 216	<b>6</b> Vulnerability 29	<b>A</b> Bug 55	Security Hotspot 31	Code Smell (101)

Tags



naming convention

Code Smell



Search by name...

Evaluating regular expressions against input strings is potentially an extremely CPU-intensive task. Specially crafted regular expressions such as (a+)+s will take several seconds to evaluate the input string

aaaaaaaaaaaaaaaaaaaaaaaaaaaaabs. The problem is that with every additional a character added to the input, the time required to evaluate the regex doubles. However, the equivalent regular expression, a+s (without grouping) is efficiently evaluated in milliseconds and scales linearly with the input size.

Evaluating such regular expressions opens the door to Regular expression Denial of Service (ReDoS) attacks. In the context of a web application, attackers can force the web server to spend all of its resources evaluating regular expressions thereby making the service inaccessible to genuine users.

This rule flags any execution of a hardcoded regular expression which has at least 3 characters and at least two instances of any of the following characters: \*+{.

Example: (a+)\*

# Ask Yourself Whether

- the executed regular expression is sensitive and a user can provide a string which will be analyzed by this regular expression.
- your regular expression engine performance decrease with specially crafted inputs and regular expressions.

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

## **Recommended Secure Coding Practices**

Check whether your regular expression engine (the algorithm executing your regular expression) has any known vulnerabilities. Search for vulnerability reports mentioning the one engine you're are using.

Use if possible a library which is not vulnerable to Redos Attacks such as  $\underline{\text{Google}}$  Re2.

Remember also that a ReDos attack is possible if a user-provided regular expression is executed. This rule won't detect this kind of injection.

## Sensitive Code Example

Django

from django.core.validators import RegexValidator
from django.urls import re\_path

RegexValidator('(a\*)\*b') # Sensitive

Comments should not be located at the end of lines of code

Code Smell

Lines should not end with trailing whitespaces

Code Smell

Files should contain an empty newline at the end

A Code Smell

Long suffix "L" should be upper case

A Code Smell

```
def define_http_endpoint(view):
    re_path(r'^(a*)*b/$', view) # Sensitive
```

re module

```
import re
from re import compile, match, search, fullmatch, split, fin

input = 'input string'
replacement = 'replacement'

re.compile('(a*)*b') # Sensitive
re.match('(a*)*b', input) # Sensitive
re.search('(a*)*b', input) # Sensitive
re.fullmatch('(a*)*b', input) # Sensitive
re.split('(a*)*b', input) # Sensitive
re.findall('(a*)*b', input) # Sensitive
re.finditer('(a*)*b', input) # Sensitive
re.sub('(a*)*b', replacement, input) # Sensitive
re.subn('(a*)*b', replacement, input) # Sensitive
```

regex module

```
import regex
from regex import compile, match, search, fullmatch, split,
input = 'input string'
replacement = 'replacement'
regex.subf('(a*)*b', replacement, input) # Sensitive
regex.subfn('(a*)*b', replacement, input) # Sensitive
regex.splititer('(a*)*b', input) # Sensitive
regex.compile('(a*)*b') # Sensitive
regex.match('(a*)*b', input) # Sensitive
regex.search('(a*)*b', input) # Sensitive
regex.fullmatch('(a*)*b', input) # Sensitive
regex.split('(a*)*b', input) # Sensitive
regex.findall('(a*)*b', input) # Sensitive
regex.finditer('(a*)*b',input) # Sensitive
regex.sub('(a*)*b', replacement, input) # Sensitive
regex.subn('(a*)*b', replacement, input) # Sensitive
```

#### Exceptions

Some corner-case regular expressions will not raise an issue even though they might be vulnerable. For example: (a | aa) +, (a | a?) +.

It is a good idea to test your regular expression if it has the same pattern on both side of a " $\mid$ ".

## See

- OWASP Top 10 2017 Category A1 Injection
- MITRE, CWE-624 Executable Regular Expression Error
- OWASP Regular expression Denial of Service ReDoS

#### Deprecated

This rule is deprecated; use {rule:pythonsecurity:S2631} instead.

Available In:



© 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