New issue

Jump to bottom

Regex DOS fixes #387

Merged **nicholasserra** merged 3 commits into `master` from `regex-dos` on Jan 20, 2021

Conversation 2 Commits 3 Checks 0 Files changed 1

nicholasserra commented on Jan 20, 2021 Collaborator

Regex DOS vulnerability report via **@b-c-ds**

- Fixes two regex via proposed patches
- Band aid on fenced-code-block DOS by limiting the amount of spaces after language specification. Passes tests and common usage.

Full report with example fixes:

~~~~
Doyensec Vulnerability Advisory
========================================================================
* Regular Expression Denial of Service (REDoS) in python-markdown2
* Affected Product: markdown2 >= 1.0.1.18
* Vendor: https://github.com/trentm
* Severity: Low
* Vulnerability Class: Denial of Service
* Status: Open
* Author(s): Ben Caller (Doyensec)
========================================================================

=== SUMMARY ===

The python package markdown2 processes markdown using regular expressions which are vulnerable to Regular Expression Denial of Service (REDoS).
If an attacker provides a malicious string, it can make markdown2 get stuck processing for a very long time.

=== TECHNICAL DESCRIPTION ===

The three vulnerable regular expressions are found in markdown2.py. They are used only when specific extras or options are enabled.

Line 894: self.regex_defns
Pattern:
            \[\#(\w+)\s* # the counter.  Open square plus hash plus a word \1
            ([^@]*)\s*   # Some optional characters, that aren't an @. \2
            @(\w+)       # the id.  Should this be normed? \3
            ([^\]]*)\]   # The rest of the text up to the terminating ] \4
Repeated character: \s
Example: '[#a' + ' ' * 3456
Enabled with the 'numbering' extra.
The section \s*([^@]*)\s* contains three infinitely repeating groups which all match space character, so a long string of spaces can cause catastrophic backtracking.
The complexity is cubic, so doubling the length of the malicious string of spaces makes processing take 8 times as long.


Line 1927: _fenced_code_block_re
Pattern:
        (?:\n+|\A\n?)
        ^```\s*?([\w+-]+)?\s*?\n    # opening fence, $1 = optional lang
        (.*?)                       # $2 = code block content
        ^```[ \t]*\n                # closing fence
Repeated character: \n
Example: '````' + '\n' * 3456
Enabled with the 'fenced-code-blocks' extra.
As the ([\w+-]+)? group is optional, the section \s*?([\w+-]+)?\s*?\n(.*?) contains three infinitely repeating groups which all match the new-line character.
Cubic complexity.


Line 535: _emacs_oneliner_vars_pat
Pattern: -\*-\s*([^\r\n]*?)\s*-\*-
Repeated character: \x20, \x09, \xa0
Example: '-*-' + ' ' * 3456
Enabled when `use_file_vars` is True.
The section \s*([^\r\n]*?)s* contains three parts which all match spaces and tabs, so a long string of spaces can cause backtracking.
Cubic complexity.

=== REPRODUCTION STEPS ===

Running any of these commands will take a while to process at 100% CPU.
Doubling the length of the repeating section will make processing take 8 times as long.

markdown2.markdown('[#a' + ' ' * 3456, extras=['numbering'])

markdown2.markdown('````' + '\n' * 3456, extras=['fenced-code-blocks'])

markdown2.markdown('-*-' + ' ' * 3456, use_file_vars=True)

=== REMEDIATION ===

Fix the three vulnerable regexes.

Potential fixes:
  regex_defns: I'd recommend replacing \s*([^@]*)\s* with ([^@]*) and then running .strip() on the result
    text_before = match.group(2).strip()

  _fenced_code_block_re is pretty hard. Line 2 could be something like:
    ^```((?!\n)\s)*?([\w+-]+((?!\n)\s)*)?\n
  using negative lookahead so the \s matches all spaces except newlines
  and moving the second \s inside group 1 so it only matches if a language is chosen.

  _emacs_oneliner_vars_pat:
    -\*-\s*(?:(\S[^\r\n]*?)([\r\n]\s*)?)?-\*-

=== DISCLOSURE TIMELINE ===
~~~~

```
2021-01-18: Vulnerability disclosed via email to maintainers

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

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.

Copyright 2021 by Doyensec LLC. All rights reserved.

Permission is hereby granted for the redistribution of this
advisory, provided that it is not altered except by reformatting
it, and that due credit is given. Permission is explicitly given
for insertion in vulnerability databases and similar, provided
that due credit is given. The information in the advisory is
believed to be accurate at the time of publishing based on
currently available information, and it is provided as-is,
as a free service to the community by Doyensec LLC. There are
no warranties with regard to this information, and Doyensec LLC
does not accept any liability for any direct, indirect, or
consequential loss or damage arising from use of, or reliance
on, this information.
```

Thanks Ben!

`Regex DOS fixes` ✗ 96dff22

nicholasserra added the **Bug** label on Jan 20, 2021

`Pretty comment alignment` ✗ e1954d3

b-c-ds commented on Jan 20, 2021

Thanks. The change looks good!
The `_fenced_code_block_re` is better but it is still quadratic because of `\s*?\n(.*?)`. It would only be a problem on super long strings of newlines now.
If you were able to do the same trick of changing `\s*` into `\s{0,99}` it should prevent that issue. Up to you.

nicholasserra commented on Jan 20, 2021 Collaborator Author

Yup, it's definitely a band aid, but in manual testing it was at least returning results quickly instead of hanging forever :P

Now that I look again, the second \s* is *normally* not used either, as the common syntax is to immediately newline, so 99 might even be overkill. Could probably just do another {0,2} to handle that and pass tests and common usage. Will experiment and maybe push another patch.

`Be forgiving` ✓ c4b4ccb

nicholasserra merged commit **7b65126** into `master` on Jan 20, 2021
2 checks passed

View details

nicholasserra deleted the `regex-dos` branch 2 years ago

Reviewers
No reviews

Assignees
No one assigned

Labels
Bug

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.

None yet

2 participants