ries Get Involved Event nties Research Advisories Get Involved Events

GHSL-2021-027: Regular expression Denial of Service in ProtonMail - CVE-2021-32816



Kevin Backhouse

Coordinated Disclosure Timeline

- 2021-01-21: Emailed report to contact@protonmail.com.

- 2021-01-21: Received auto-reply email from contact@protonmail.com.
 2021-02-25: Created an issue
 2021-02-25: Response from Serge Droz (Proton-CERT: cert@protonmail.ch). They overlooked the original email, but have now found it and are working on a fix.
- 2021-03-12: Bug is fixed.
 2021-03-13: Serge Droz confirms that the bug is fixed.

Summary

The project contains one or more regular expressions that are vulnerable to ReDoS (Regular Expression Denial of Service)

ProtonMail

Tested Version

Latest commit at the time of reporting (January 21, 2021).

Details

ReDoS

ReDoS, or Regular Expression Denial of Service, is a vulnerability affecting poorly constructed and potentially inefficient regular expressions which can make them perform extremely badly given a creatively constructed input string.

For the specific regular expression reported, it is possible to force it to work with an O(2^n) runtime performance when there is exponential backtracking,

ReDoS can be caused by ambiguity or overlapping between some regex clauses. These badly performing regular expressions can become a security issue if a user can control the input. For example if the project is an input validation library, then the project could be used by a server to validate untrusted user input. There is no one size fits all when it comes to fixing ReDoS. But in general it is about removing ambiguity/overlap inside the regular expression

Before showing the vulnerable regex, it may be helpful to show some examples of regular expressions vulnerable to ReDoS and how to fix them. If you are familiar with this vulnerability and how to fix it, please skip this section.

var reg = /<!--(.|\s)*?-->/q;

The above regular expression matches the start of an HTML comment, followed by any characters, followed by the end of a HTML comment. The dot in the regular expression (.) matches any char except newlines, and \(\chi_8\) matches any whitespace. Both and \s matches whitespace such as the space character. There are therefore many possible ways for this regular expression to match a sequence of spaces. This becomes a problem if the input is a string that starts with <!-- followed by a long sequence of spaces, because the regular expression evaluator will try every possible way of matching the spaces (see this debugging session for an example: https://regex101.com/r/XvYgkN/1/debugger).

The fix is to remove the ambiguity, which can be done by changing the regular expression to the below, where there is no overlap between the different elements of the regular expression

var reg = /<!--(.|\r|\n)*?-->/g;

var reg = /(\w+_?)+_(\d+)/;

The above matches a snake_case identifier that ends with some digits. However, for a string starting with lots of letters, there many ways for the regular expression to match those letters, due to the regex having a repetition matching letters ((w+) inside another repetition that can match letters ((\wmu+\frac{1}{2})+). The regular expression evaluator will try every possible grouping of letters into smaller groups of letters (see this debugging session for an example: https://regex101.com/r/fmci1j/1/debugger). The fix is again to remove ambiguity, by changing the inner repetition to match groups of letters that must end with an underscore.

Often the regular expressions are not as simple as the examples above. Like the below regular expression that used to be part of VS Code. (the top regular expression is the vulnerable, the bottom is the fixed)

But this example is actually very similar to the first example. A section of the regular expression that was too general (\s) was changed to something a bit more specific ([\s\s\/\)] to remove overlap with another part of the regular expression.

Vulnerability

ProtonMail is a mail client for handling encrypted emails. The method for extracting public keys from a message is vulnerable to exponential-redos

This vulnerability was found using a CodeQL query which identifies inefficient regular expressions

For simplicity, the PoC has copy-pasted the relevant regular expressions.

· Run the below with node

var reg3 = $/^{(:\cdot)s^{(:,s)}})$ s*=\s*[^;\s]*\s*keydata\s*=([^;]*)\$/; reg3.test(attackString); // <- doesn't terminate

Impact

This issue may lead to a denial of service.

CVE

• CVE-2021-32816

Credit

This issue was discovered and reported by GitHub team member @erik-krogh (Erik Krogh Kristensen).

Contact

You can contact the GHSL team at securitylab@github.com, please include a reference to GHSL-2021-027 in any communication regarding this issue.

GitHub

Product

- Features
 Security
 Enterprise
 Customer st
 Pricing
 Resources

Platform

- Developer API
 Partners
 Atom
 Electron
 GitHub Desktop

Support

Company

- About
 Blog
 Careers
 Press
 Shop

 f

 6

 2021 GitHub, Inc.
 Terms
 Privacy
 Cookie settings