GHSL-2020-358-redos-Schema-Inspector.md

sec.md

GitHub Security Lab (GHSL) Vulnerability Report: `GHSL-2020-358`

The GitHub Security Lab team has identified potential security vulnerabilities in Schema-Inspector.

We are committed to working with you to help resolve these issues. In this report you will find everything you need to effectively coordinate a resolution of these issues with the GHSL team.

If at any point you have concerns or questions about this process, please do not hesitate to reach out to us at `securitylab@github.com` (please include `GHSL-2020-358` as a reference).

If you are *NOT* the correct point of contact for this report, please let us know!

Summary

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

Product

Schema-Inspector

Tested Version

Latest commit at the time of reporting (December 21, 2020).

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)*?-->/g;
```

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 `\s` 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 (`(\w+_?)+`). 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.

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

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)

```
var linkPattern = /(\[((!\[[^\]]*?\]\(\s*)([^\s\(\)]+?)\s*\)\]|(?:\\\]|[^\]])*\])\(\s*)(([^\s\(\)]|\(\S*?\))+)\s*(".*?")
var linkPattern = /(\[((!\[[^\]]*?\]\(\s*)([^\s\(\)]+?)\s*\)\]|(?:\\\]|[^\]])*\])\(\s*)(([^\s\(\)]|\([^\s\(\)]*?\))+)\s*
```

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\(\)]`) to remove overlap with another part of the regular expression.

Vulnerability

Schema-Inspector is a JSON schema validation/parsing library.

This issue was detected using the following CodeQL query.

PoC:

- Open this demo page: http://atinux.github.io/schema-inspector/
- Paste the below text into the `Validation` textbox.

```
{"type": "object","properties": {"email": { "type": "string", "pattern": "email" }}}
```

- Paste the below text into the `Data` textbox.

```
{"email": "a@0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0."}
```

Impact

This issue may lead to a denial of service.

GitHub Security Advisories

We recommend you create a private GitHub Security Advisory for these findings. This also allows you to invite the GHSL team to collaborate and further discuss these findings in private before they are published.

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-2020-358` in any communication regarding this issue.

Disclosure Policy

This report is subject to our coordinated disclosure policy.