SYNOPSYS®
Silicon to Software™

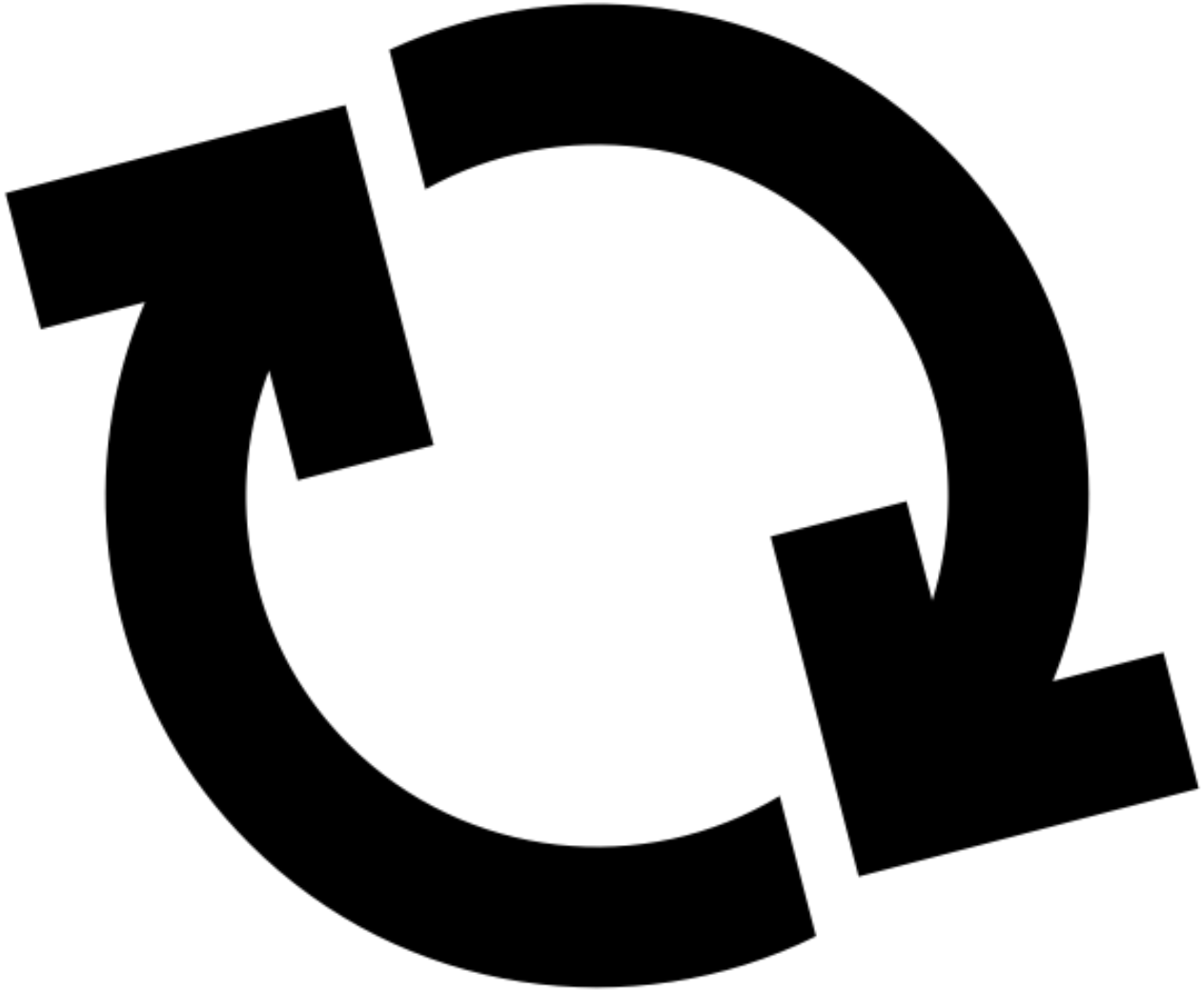
This DoS Goes Loop-di-Loop

Allon Mureinik
Senior Manager, Seeker Agents
Synopsys, Inc.
allon.mureinik@synopsys.com / @mureinik / https://www.linkedin.com/in/mureinik/
FOSDEM, 06/02/2021

Quick Shoutout

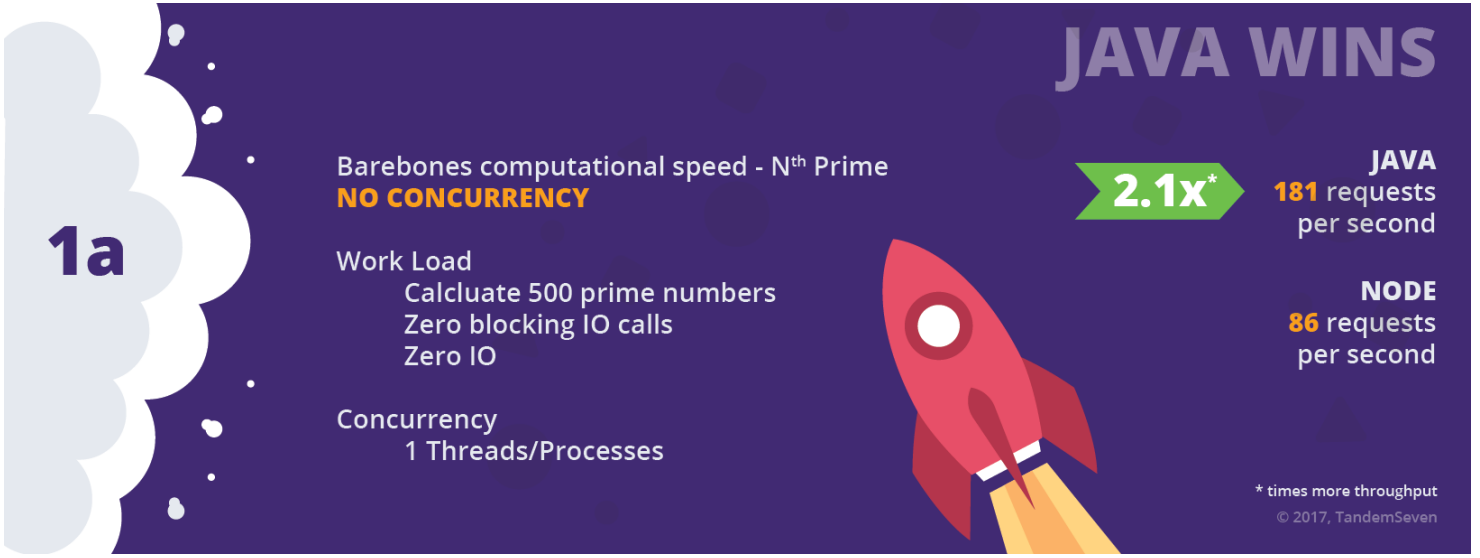
https://telaviv.appsecglobal.org

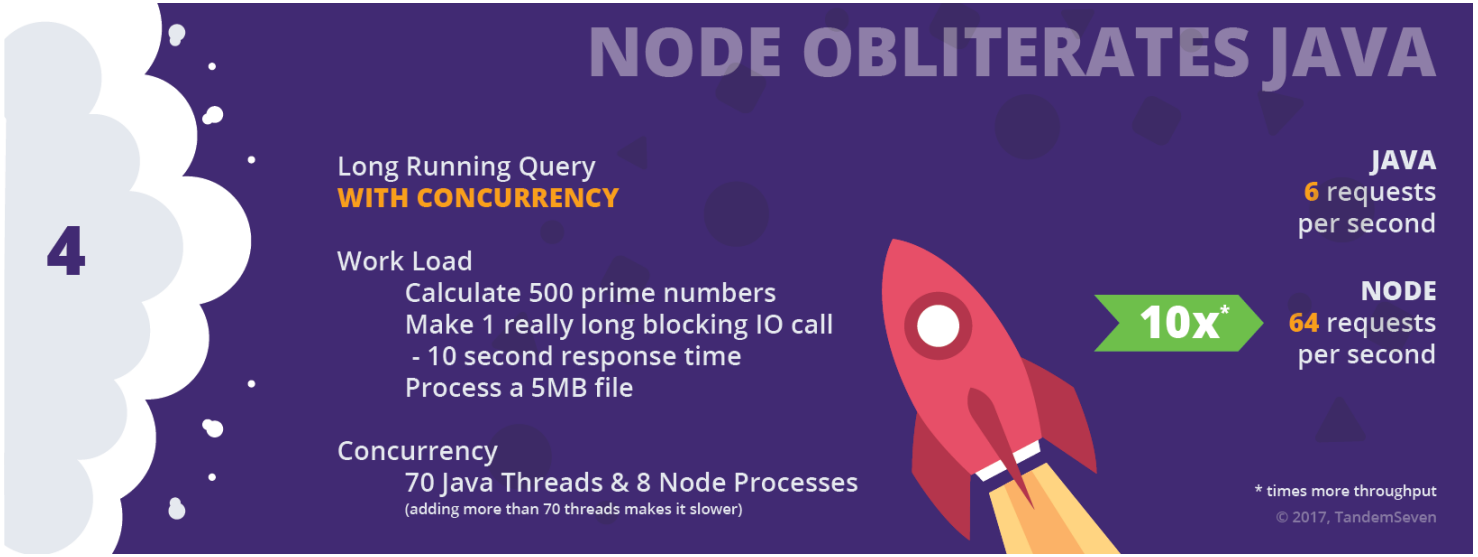
Quick Reminder: Node.js' Event Loop



https://thenounproject.com/term/redo/62716

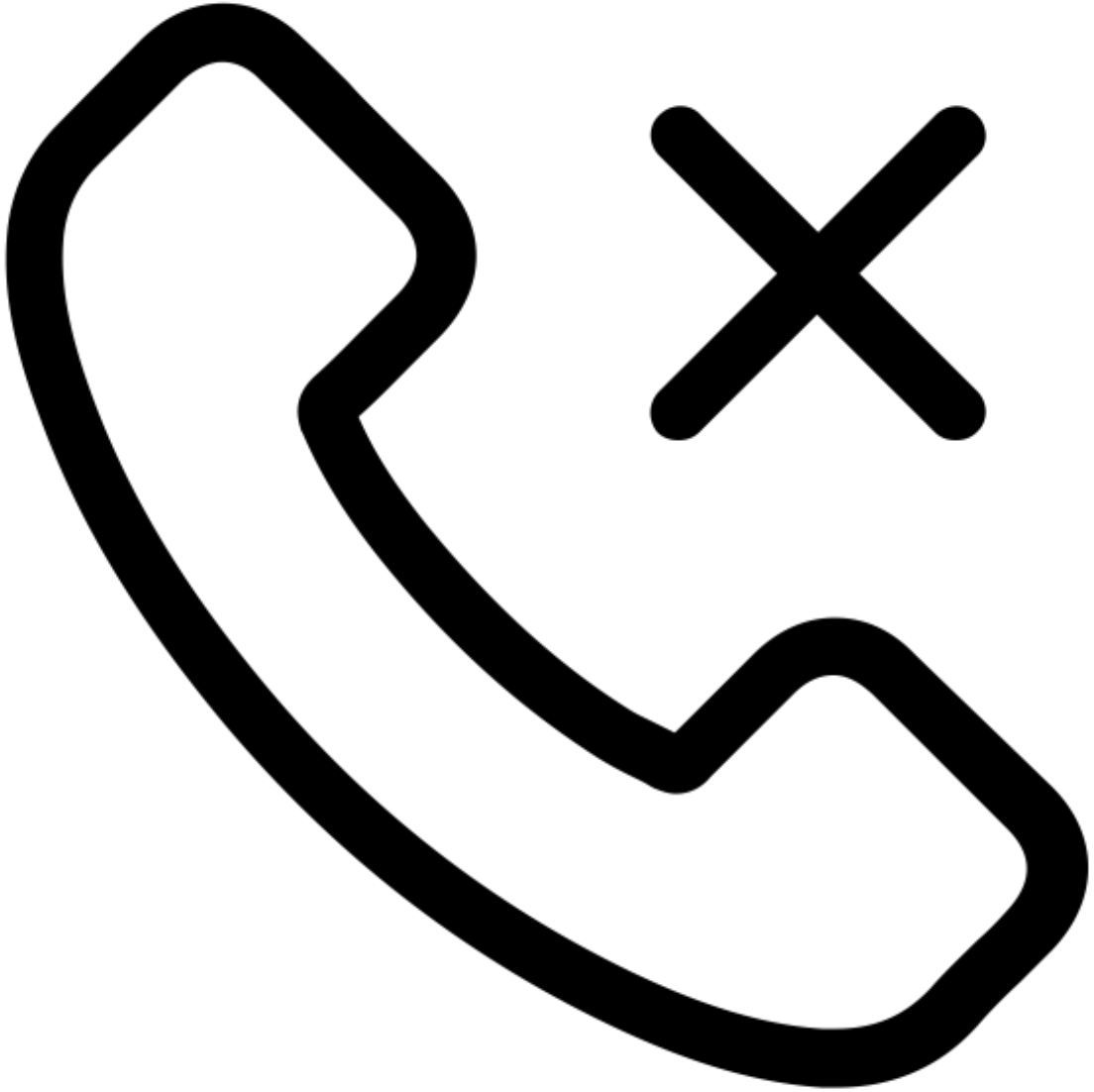
Benchmarks





https://www.tandemseven.com/blog/performance-java-vs-node/

Reminder: Denial of Service (DoS)



https://thenounproject.com/term/decline/373722

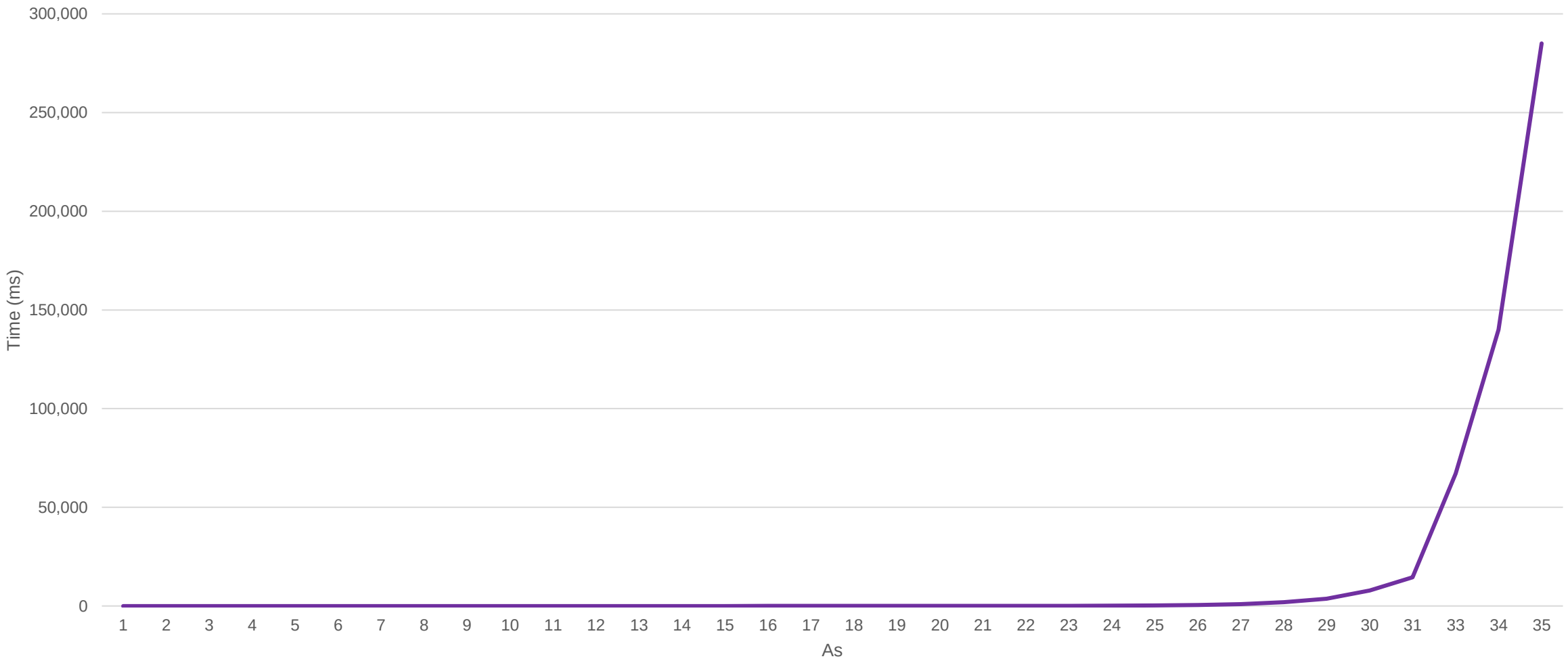
Regex DoS (ReDoS)

```
const express = require('express');
const app = express();

app.get('/regexp', (req, res) => {
    const regexp = new RegExp(req.query.regexp);
    const text = req.query.text;
    res.end(regexp.test(text) ? 'Match!' : 'No match');
});

app.listen(3000, () => console.log('Listening on port 3000'));
```

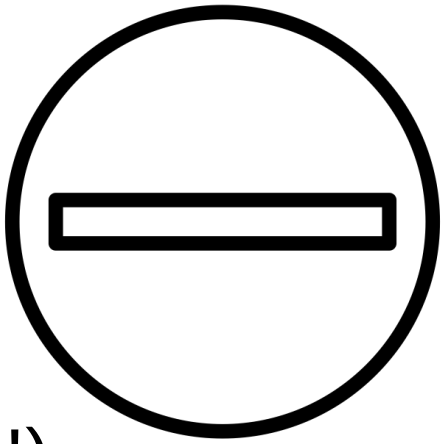
Regex DoS (ReDoS) – how bad is it really?

```
console.log('A\'s\ttime');
let regex = new RegExp('^(a+)+$');

for (let i = 1; i < 100; ++i) {
    const str = Array(i + 1).join('a') + 'b';
    const before = new Date();
    regex.test(str);
    const after = new Date();
    const time = after - before;
    console.log(`${i}\t${time}`);
}
```

Regex DoS (ReDoS) – results



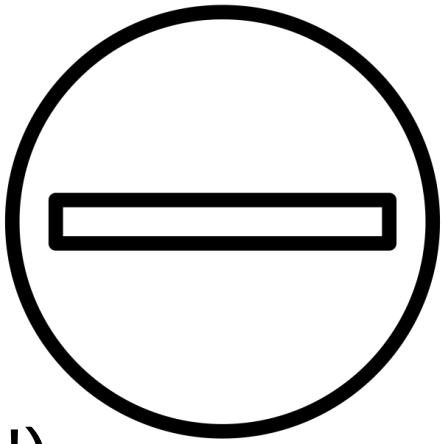
Regex DoS (ReDoS) – Remediation



- Don’t allow tainted input as regex
  - Not always possible…
  - If you must, sanitize it (safe-regex to the rescue!)

https://thenounproject.com/term/no-entry/980838

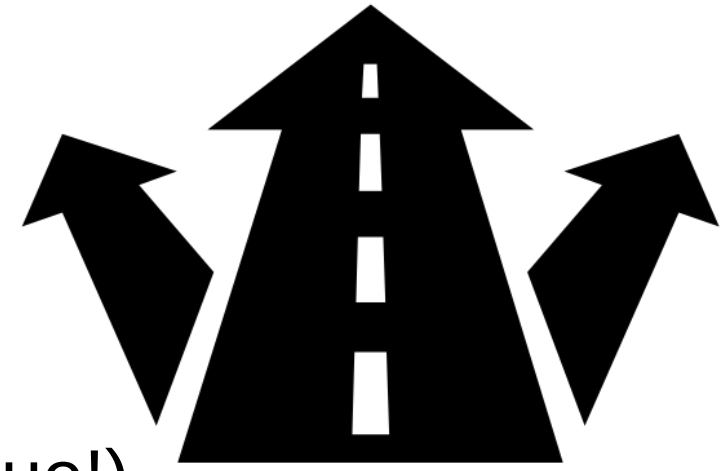
Regex DoS (ReDoS) – Remediation



- Don’t allow tainted input as regex
  - Not always possible…
  - If you must, sanitize it (safe-regex to the rescue!)
- Don’t allow tainted input to be evaluated by a dodgy regex
  - Usually not possible…
  - Use length limits

https://thenounproject.com/term/no-entry/980838

Regex DoS (ReDoS) – Remediation



- Don’t allow tainted input as regex
  - Not always possible…
  - If you must, sanitize it (safe-regex to the rescue!)
- Don’t allow tainted input to be evaluated by a dodgy regex
  - Usually not possible…
  - Use length limits
- Think about alternative solutions
  - re2 isn’t vulnerable to ReDoS
  - Use specific tools for specific needs (e.g., validator.js)

https://thenounproject.com/term/alternative-route/2902159

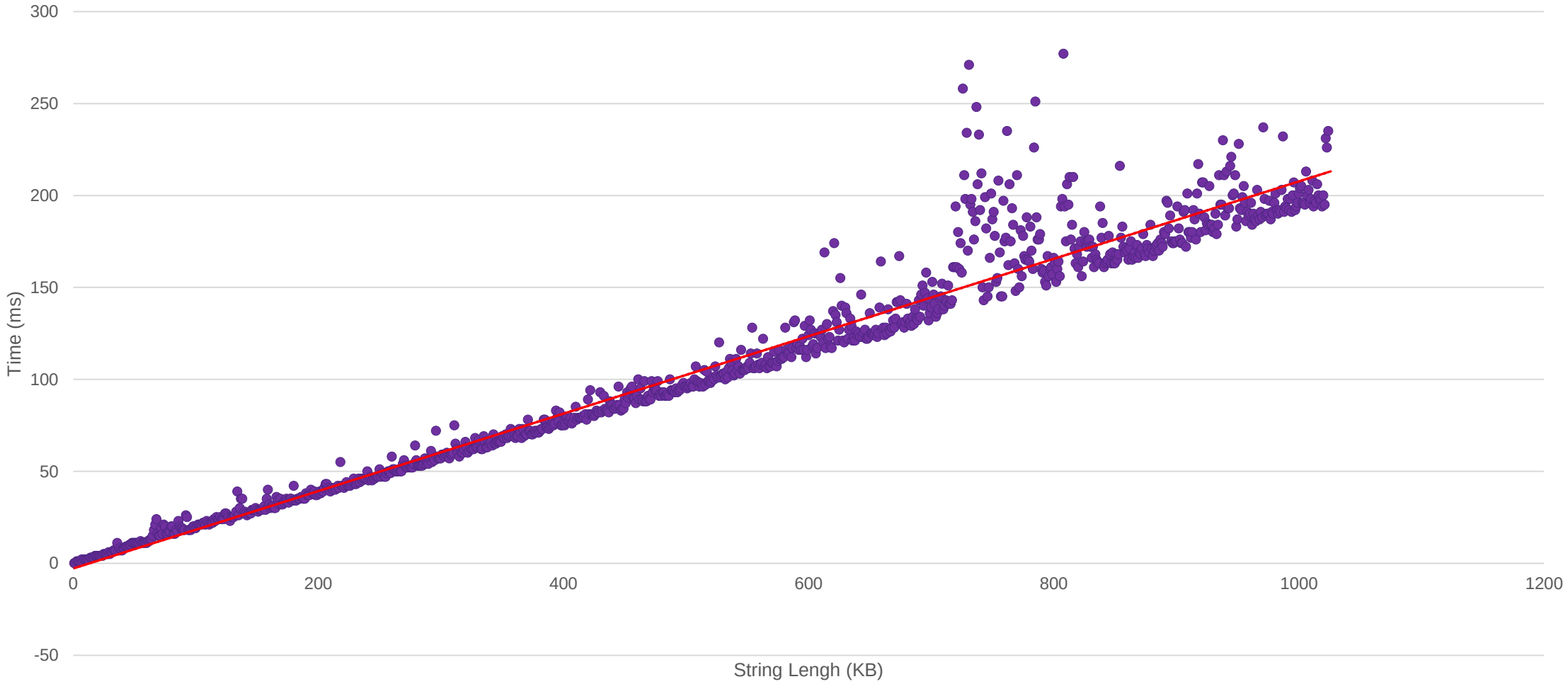
JSON DoS

```
const express = require('express');
const app = express();
app.use(express.json());

app.post('/json', (req, res) => {
   res.end(Object.keys(req.body).length + ' keys in the payload');
});

app.listen(3000, () => console.log('Listening on port 3000'));
```

JSON DoS – How bad is it really?

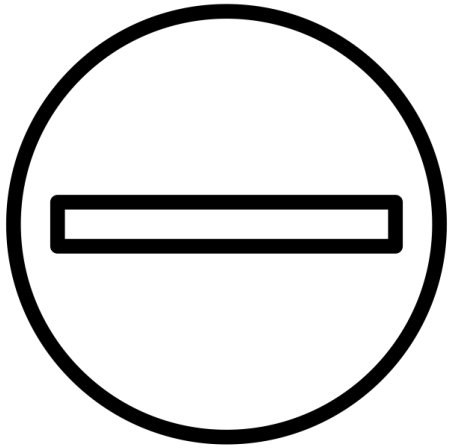
```
console.log('Length\tTime');
for (let i = 1024; i <= 1024 * 1024 ; i += 1024) {
    const str = "" + Array(i + 1).join('a') + "";
    const before = new Date();
    for (let j = 1; j < 100; ++j) {
        JSON.parse(str);
    }
    const after = new Date();
    console.log(`${i}\t${after-before}`);
}
```

JSON DoS – Results



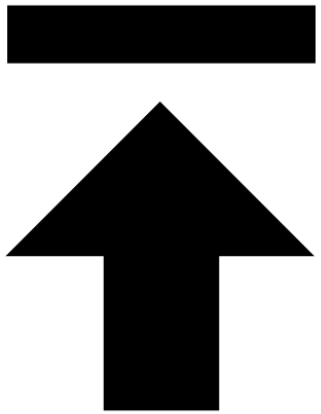
JSON DoS – Remediation

- Don’t allow tainted input to be parsed
  - Not realistic…



https://thenounproject.com/term/no-entry/980838

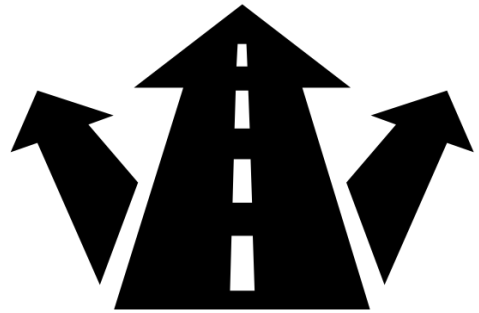
JSON DoS – Remediation



- Don’t allow tainted input to be parsed
  - Not realistic…
- Limit the size of the input
  - Express: `app.use(express.json({limit: '50kb'})`
  - Hapi: `route.options.payload.maxBytes = 50 * 1024`
  - Fastify: `Fastify({bodyLimit: 50 * 1024})`

https://thenounproject.com/term/top/446421

JSON DoS – Remediation



- Don’t allow tainted input to be parsed
  - Not realistic…
- Limit the size of the input
  - Express: `app.use(express.json({limit: '50kb'})`
  - Hapi: `route.options.payload.maxBytes = 50 * 1024`
  - Fastify: `Fastify({bodyLimit: 50 * 1024})`
- If you aren’t parsing JSON by a middleware, consider alternative libraries like `BFJ` or `JSONStream`

https://thenounproject.com/term/alternative-route/2902159/

Storage (I/O) DoS

```
const fs = require('fs');
const path = require('path');
const express = require('express');
const app = express();

app.get('/lorem', (req, res) => {
   res.end(fs.readFileSync(path.join(__dirname, 'lorem.txt')));
});

app.listen(3000, () => console.log('Listening on port 3000'));
```

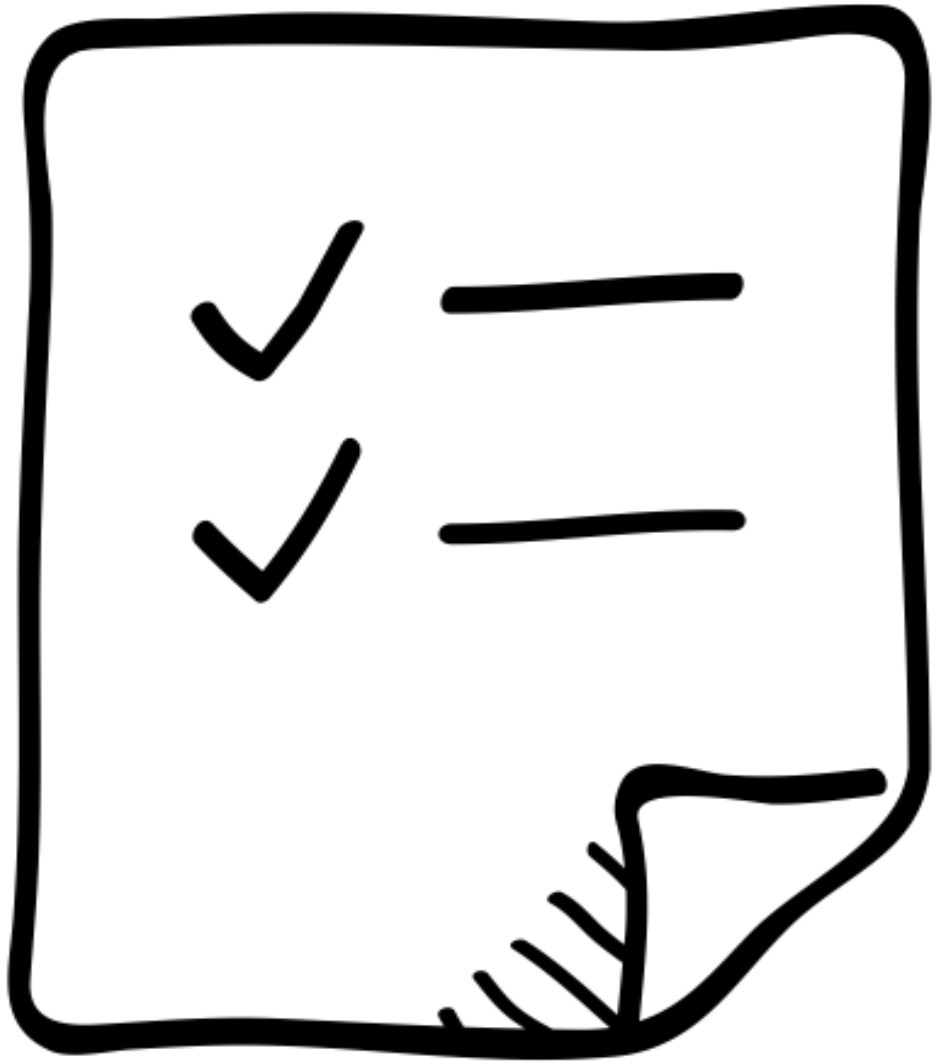
Storage (I/O) DoS – Remediation

The are two ways to perform storage operations in Node.js:

1. The async way
   - Delegate a storage operation to the kernel, and wait for a callback
   - E.g.: `fs.readDir`, `fs.writeFile`, etc
   - 3rd parties follow similar patterns (e.g., `fs-extra`, `adm-zip`)

2. The **wrong** way

Summary



https://thenounproject.com/term/brief/935656

Summary

- Any code that relies on tainted input should have some sort of sanitation/protection

- If a function has a variant with a callback – use it
  - This isn't limited to I/O (e.g., `crypto.randomBytes`)

- Test!
  - Make security and stress tests part of you CI/CD pipeline
  - Incorporate tools that can highlight these issues

Some links

- Don’t Block the Event Loop: https://nodejs.org/en/docs/guides/dont-block-the-event-loop

- Demos and Benchmarks: https://github.com/mureinik/loop-li-loop

Questions?

https://thenounproject.com/term/questions/1195076/

SYNOPSYS®
Silicon to Software™

Thank You

Contact
allon.mureinik@synopsys.com
@mureinik
https://www.linkedin.com/in/mureinik/