Pragmatic Web Security

Security for developers

Building secure React applications

@PhilippeDeRyck

Dr. Philippe De Ryck

What Is the React.js Framework? When and Why Should I Use React.js in My Project?

Piotr Balbier

Jul 2, 2019 | 14 min read Web Development React.js

For building web and mobile apps, React.js has it all.

It's fast, secure, and scalable. It provides a fantastic user and developer experience. What's more, its popularity is growing as it's supported by Facebook and a vibrant community. If you are wondering which technology to choose for your project, React.js is probably the best choice. Here's a short guide that will explain why.

Dr. Philippe De Ryck

- Deep understanding of the web security landscape
- Google Developer Expert (not employed by Google)
- Course curator of the SecAppDev course
  (https://secappdev.org)

Pragmatic Web Security

High-quality security training for developers and managers

Custom courses covering web security, API security, Angular security, …

Consulting services on security, OAuth 2.0, OpenID Connect, …

@PhilippeDeRyck

https://PragmaticWebSecurity.com

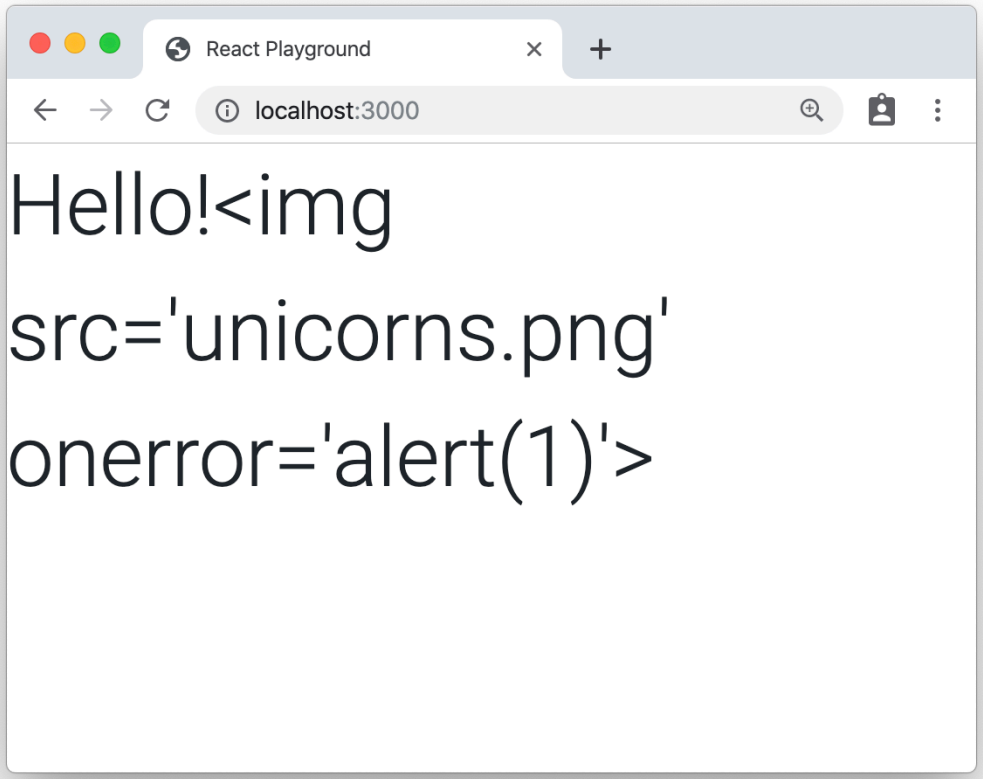
REACT CODE

```
return ( <div>{ data }</div> );
```

UNTRUSTED DATA

```
Hello!
<img src='unicorns.png' onerror='alert(1)'>
```

RENDERED PAGE



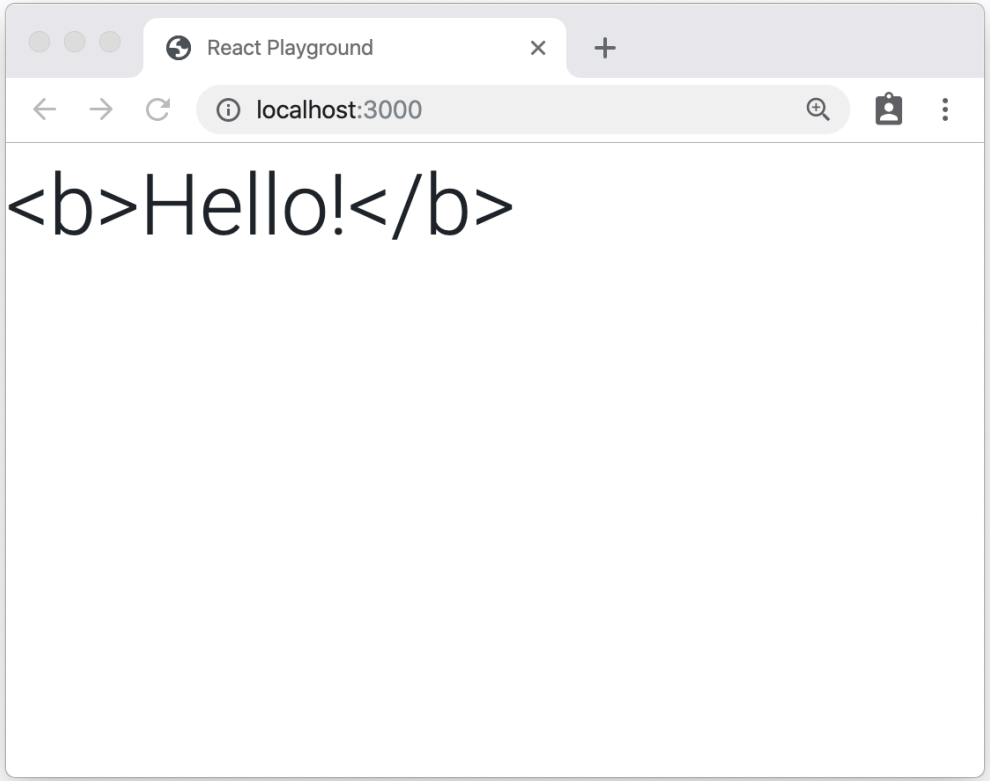
REACT CODE

```
return ( <div>{ data }</div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
```

RENDERED PAGE



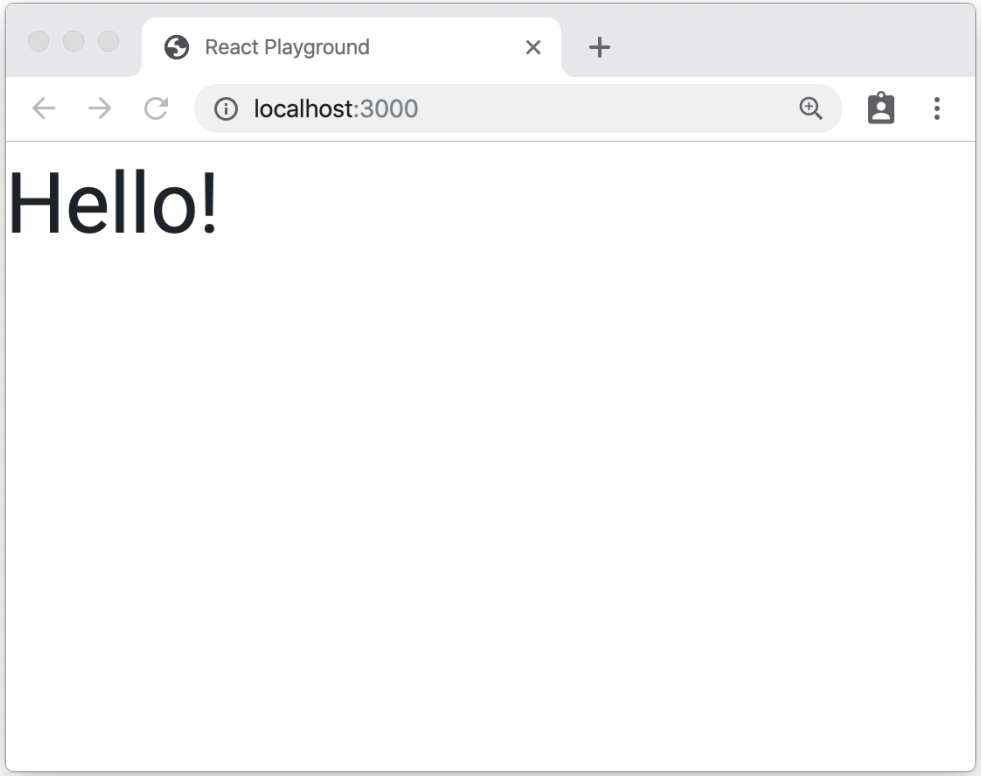
REACT CODE

```
return (
<div dangerouslySetInnerHTML={{__html: data}}>
</div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
```

RENDERED PAGE



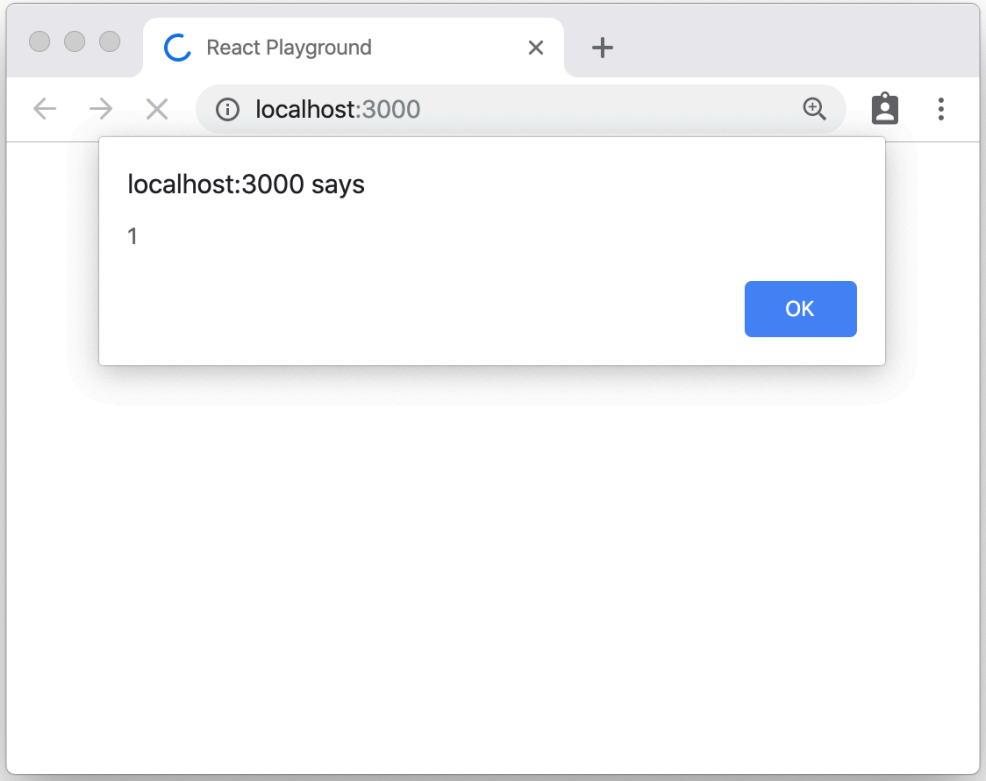
REACT CODE

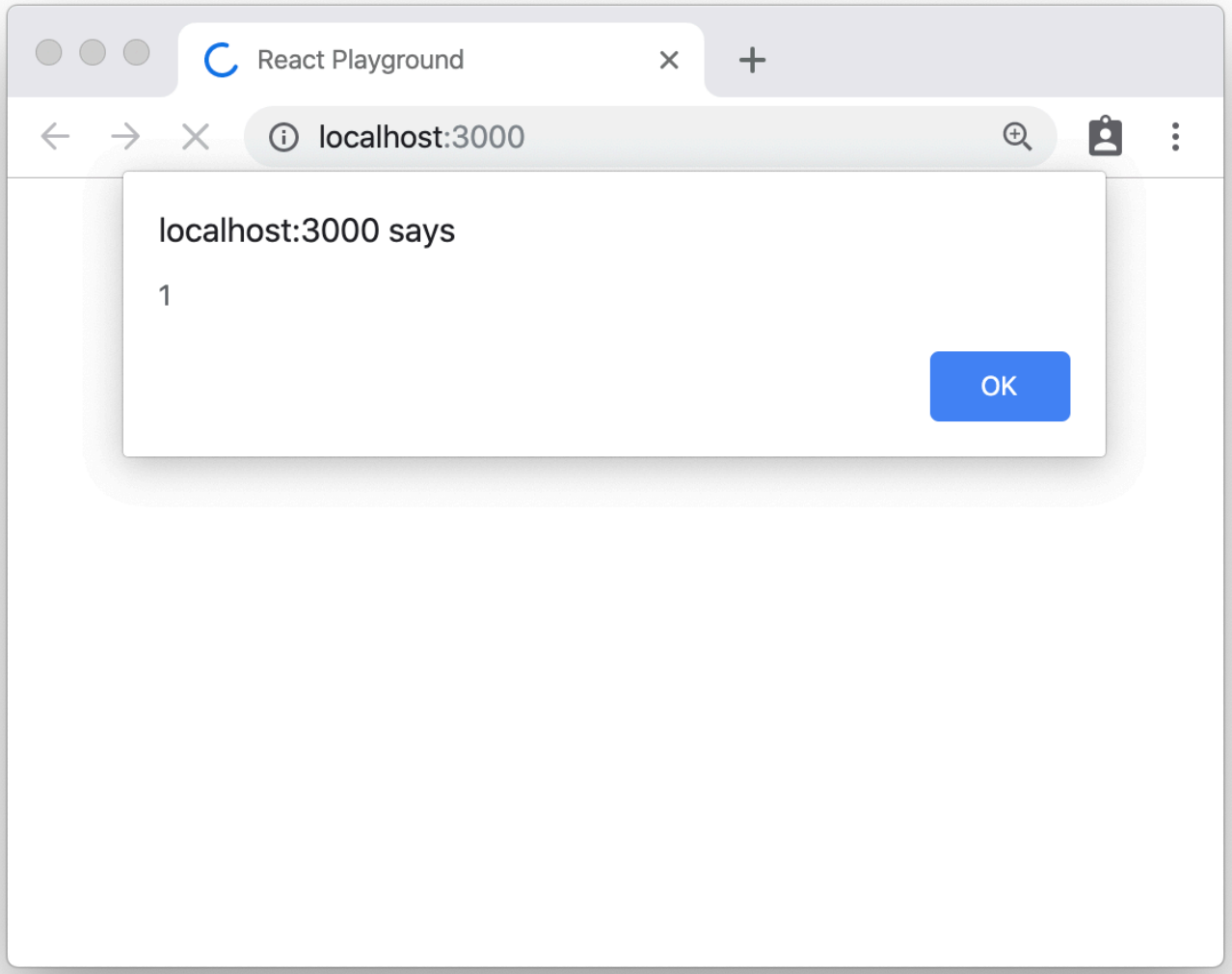
```
return (
<div dangerouslySetInnerHTML={{__html: data}}>
</div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
<img src='unicorns.png' onerror='alert(1)'>
```

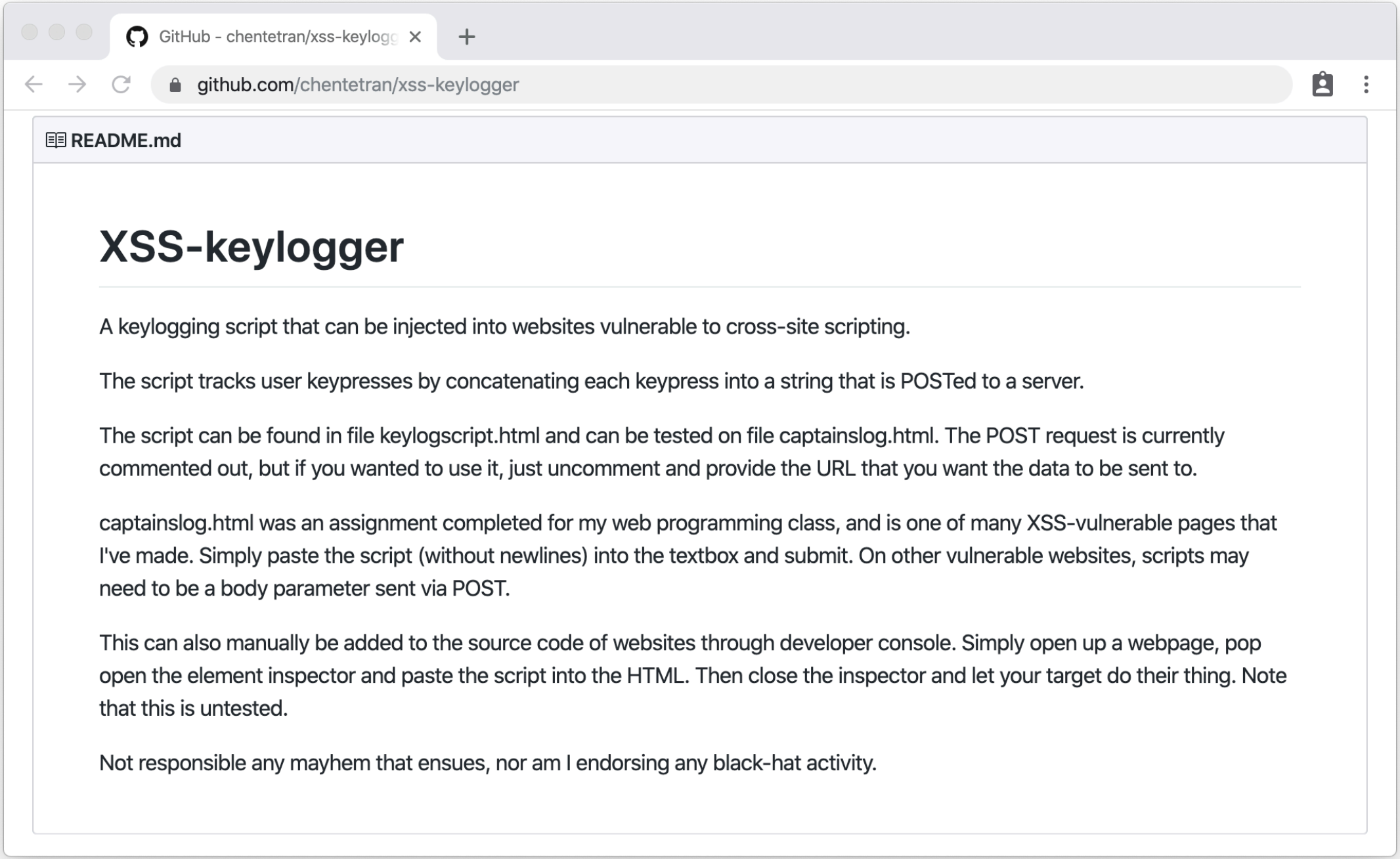
RENDERED PAGE

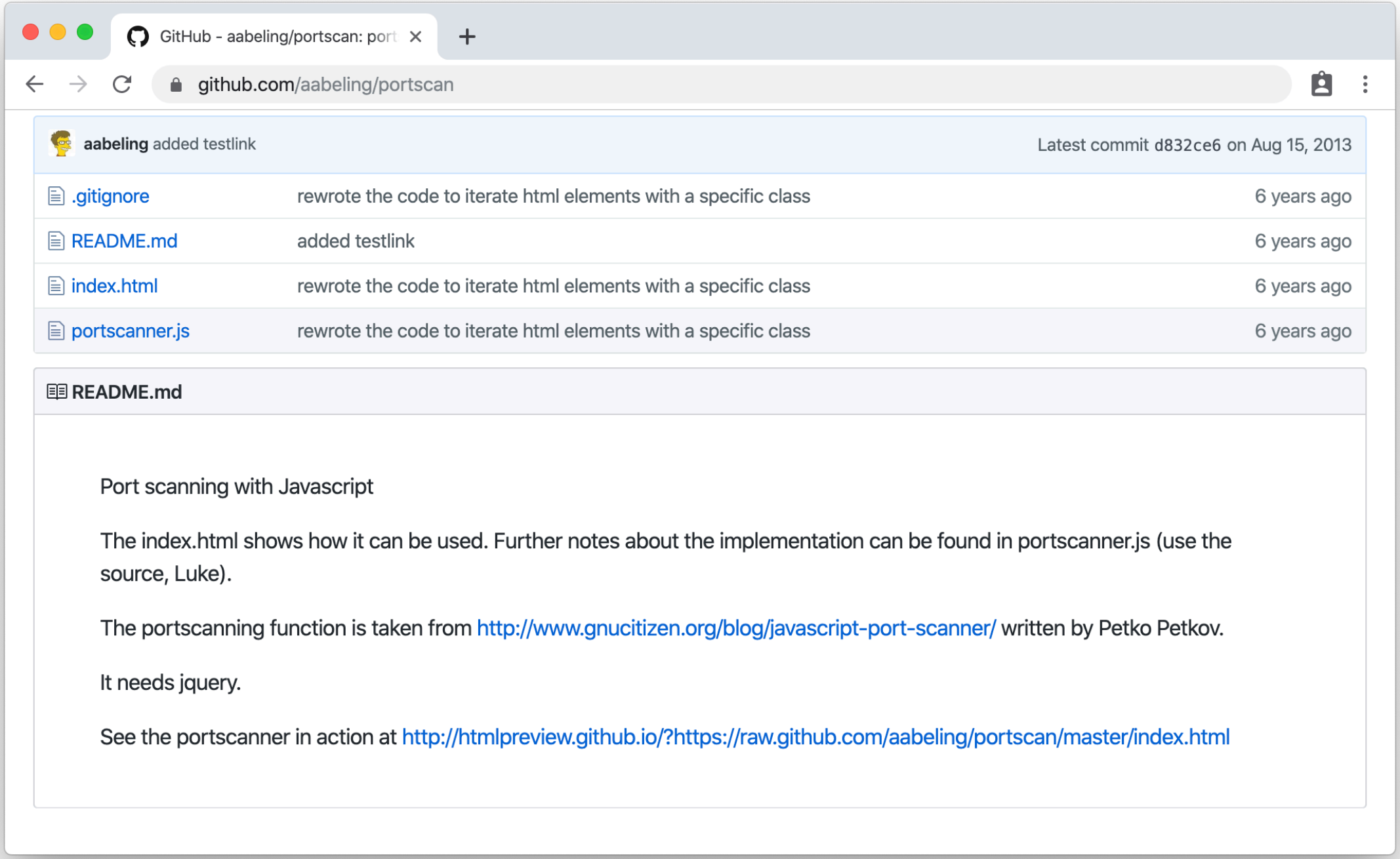



React Playground
localhost:3000
localhost:3000 says
1
OK

Hy Admin,
Just want to tell you this site is not secure
Please Fix it As Soon As Possible (ASAP)
I have to go now
Just Remember We Will Come Again
Bye ..
Peace Contect fb Me:
[KLIK DISINI]

PLEASE UPGRADE YOUR SECURITY


GitHub - chentetran/xss-keylogg
github.com/chentetran/xss-keylogger
README.md
XSS-keylogger
A keylogging script that can be injected into websites vulnerable to cross-site scripting.
The script tracks user keypresses by concatenating each keypress into a string that is POSTed to a server.
The script can be found in file keylogscript.html and can be tested on file captainslog.html. The POST request is currently commented out, but if you wanted to use it, just uncomment and provide the URL that you want the data to be sent to.
captainslog.html was an assignment completed for my web programming class, and is one of many XSS-vulnerable pages that I've made. Simply paste the script (without newlines) into the textbox and submit. On other vulnerable websites, scripts may need to be a body parameter sent via POST.
This can also manually be added to the source code of websites through developer console. Simply open up a webpage, pop open the element inspector and paste the script into the HTML. Then close the inspector and let your target do their thing. Note that this is untested.
Not responsible any mayhem that ensues, nor am I endorsing any black-hat activity.


GitHub - aabeling/portscan: port
github.com/aabeling/portscan
aabeling added testlink
Latest commit d832ce6 on Aug 15, 2013
.gitignore
rewrote the code to iterate html elements with a specific class
6 years ago
README.md
added testlink
6 years ago
index.html
rewrote the code to iterate html elements with a specific class
6 years ago
portscanner.js
rewrote the code to iterate html elements with a specific class
6 years ago
README.md
Port scanning with Javascript
The index.html shows how it can be used. Further notes about the implementation can be found in portscanner.js (use the source, Luke).
The portscanning function is taken from http://www.gnucitizen.org/blog/javascript-port-scanner/ written by Petko Petkov.
It needs jquery.
See the portscanner in action at http://htmlpreview.github.io/?https://raw.github.com/aabeling/portscan/master/index.html

£ 183 million

British Airways faces a massive fine after the Magecart attack

Signal Messenger

```
@@ -111,7 +113,9 @@ export class Quote extends React.Component<Props, {}> {
```

```
111
112         if (text) {
113           return (
-114             <div className="text" dangerouslySetInnerHTML={{ __html: text }} />
115           );
116         }
117
```

```
113
114         if (text) {
115           return (
+116             <div className="text">
+117               <MessageBody text={text} />
+118             </div>
119           );
120         }
121
```

dangerouslysetinnerhtml

Search

- Repositories 35
- **Code 358K**
- Commits 8K
- Issues 4K
- Packages 0
- Marketplace 0
- Topics 1
- Wikis 130
- Users 0

Languages

JavaScript	242,692
JSX	44,537
TypeScript	14,708

358,376 code results

Sort: Best match

chandlerzhang/touch-60824-redux

src/components/SeatMap.js

```
53                                  <td id="6" dangerouslySetInnerHTML=
54                                      {{__html: m.value_6}}/>
55                                  <td id="12" dangerouslySetInnerHTML=
56                                      {{__html: m.value_12}}/>
```

JavaScript Showing the top two matches Last indexed on Jul 4, 2018

kumuluz/kumuluz.io

src/content/events.js

```
22          <span>
23            <p dangerouslySetInnerHTML={{__html:
    t('timeline.events.oct2013.content1')}} />
24            <img src={imgEvent2013} alt={t('timeline.events.oct2013.title')}
    className="img-fluid my-3 d-block mx-auto" />
25            <p dangerouslySetInnerHTML={{__html:
    t('timeline.events.oct2013.content2')}} />
26          </span>
```

JavaScript Showing the top two matches Last indexed on Dec 12, 2018

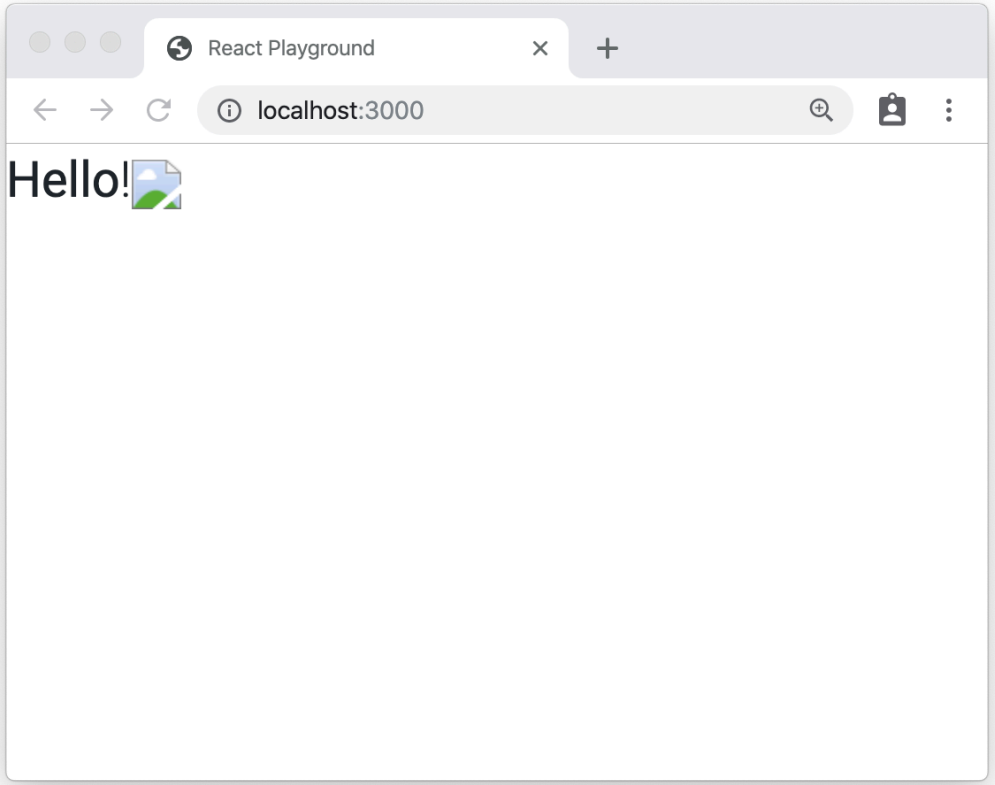
REACT CODE

```
return (
<div dangerouslySetInnerHTML={{__html:
DOMPurify.sanitize(data)}}></div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
<img src='unicorns.png' onerror='alert(1)'>
```

RENDERED PAGE



REACT CODE

```
return (
<div dangerouslySetInnerHTML={{__html:
DOMPurify.sanitize(data)}}></div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
<img src='unicorns.png' onerror='alert(1)'>
```

THE DOM

```
<b>Hello</b>!
<img src='unicorns.png' onerror='alert(1)'>
```


```
const createDOMPurify = require('dompurify');
const DOMPurify = createDOMPurify(window);


const Application = () => {
  let data = "Hello!<img src='unicorns.png' onerror='alert(1)'>";
  return ( <div dangerouslySetInnerHTML=
                    {{__html: DOMPurify.sanitize(data)}}>
           </div> );
};
```

Avoiding XSS in React

Use simple data binding in JSX

Only use dangerouslySetInnerHTML in combination with sanitization

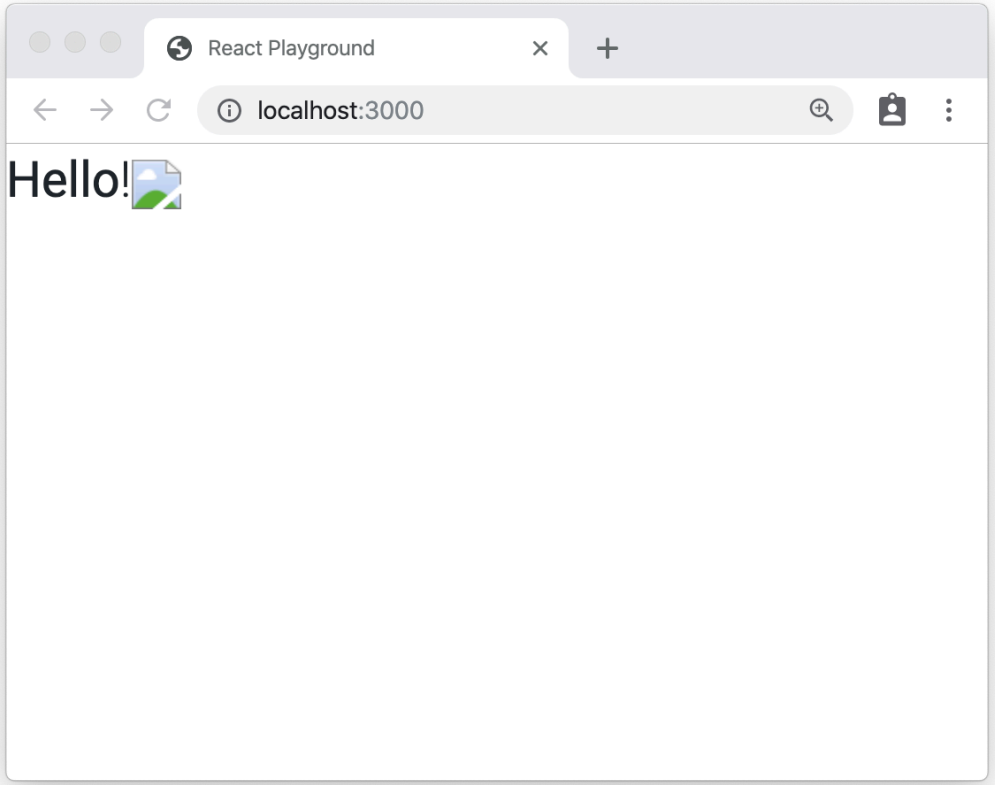
REACT CODE

```
return ( <div> { ReactHtmlParser(data) }</div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
<img src='unicorns.png' onerror='alert(1)'>
```

RENDERED PAGE



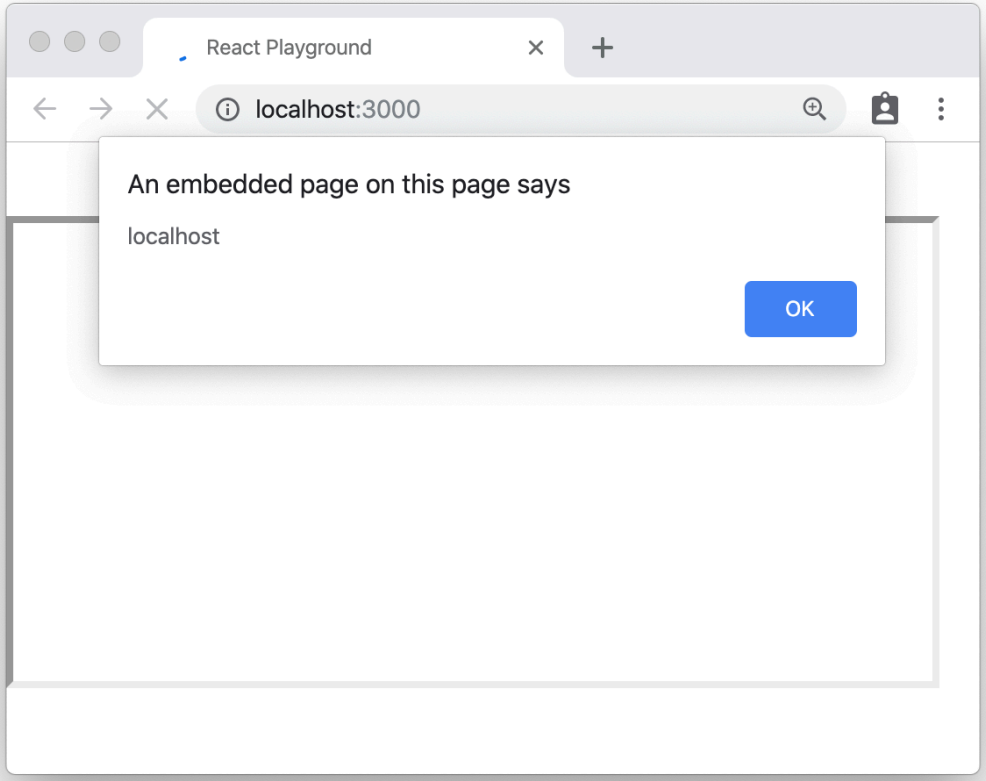
REACT CODE

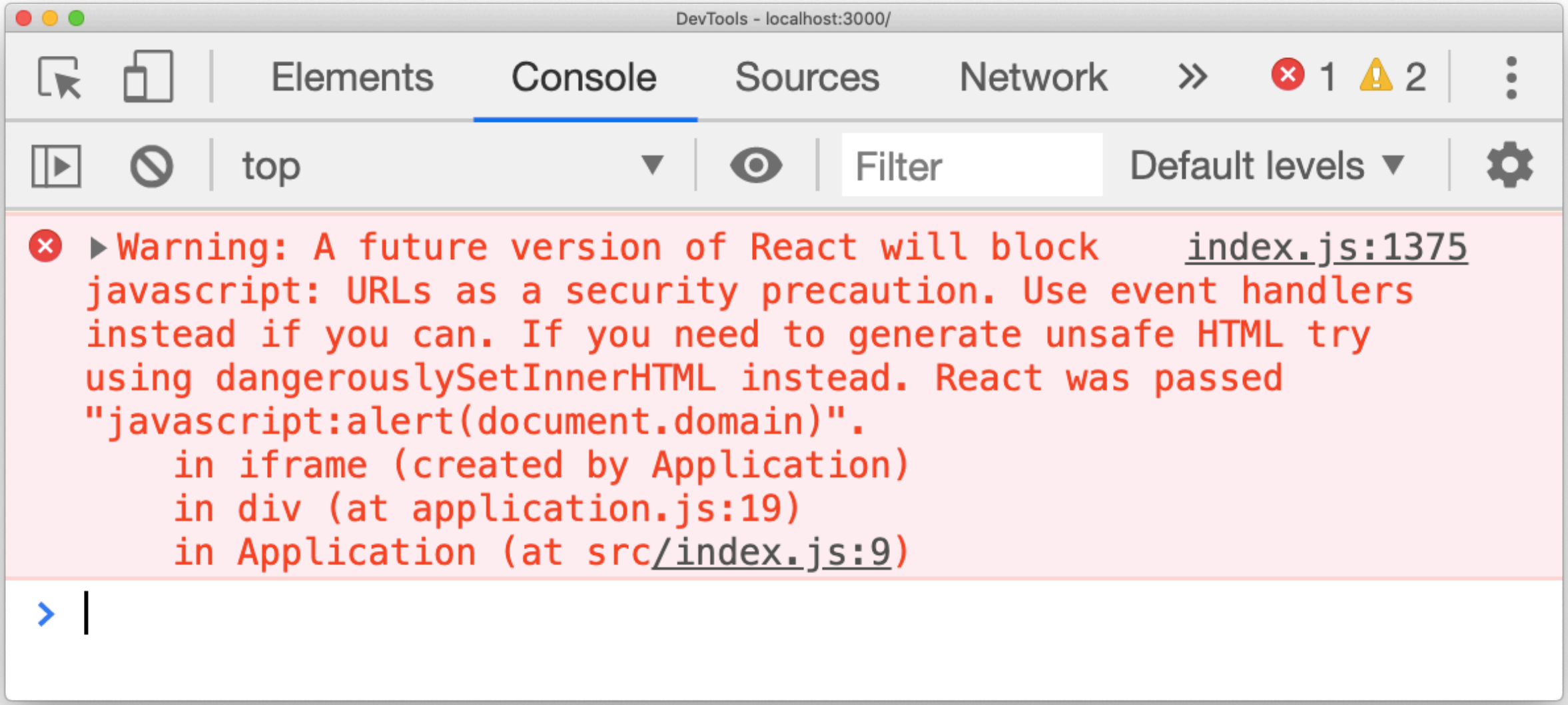
```
return ( <div> { ReactHtmlParser(data) }</div> );
```

UNTRUSTED DATA

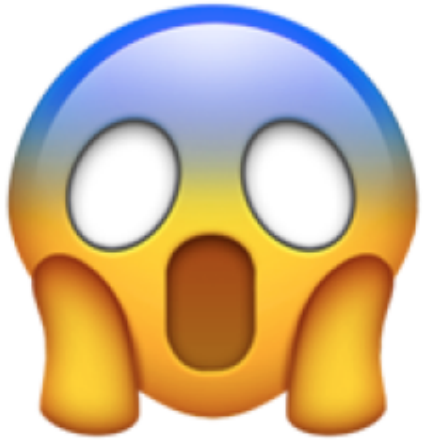
```
<b>Hello</b>!
<iframe src='javascript:alert(document.domain)'>
```

RENDERED PAGE




DevTools - localhost:3000/
Elements
Console
Sources
Network
1
2
top
Filter
Default levels
▶Warning: A future version of React will block javascript: URLs as a security precaution. Use event handlers instead if you can. If you need to generate unsafe HTML try using dangerouslySetInnerHTML instead. React was passed "javascript:alert(document.domain)".
in iframe (created by Application)
in div (at application.js:19)
in Application (at src/index.js:9)
index.js:1375

```
const isJavaScriptProtocol = /^[\u0000-
\u001F ]*j[\r\n\t]*a[\r\n\t]*v[\r\n\t]*a
[\r\n\t]*s[\r\n\t]*c[\r\n\t]*r[\r\n\t]*i
[\r\n\t]*p[\r\n\t]*t[\r\n\t]*\:/i;
```



https://github.com/facebook/react/blob/103378b1eada44561821b1c22ff54e0537cf9764/packages/react-dom/src/shared/sanitizeURL.js

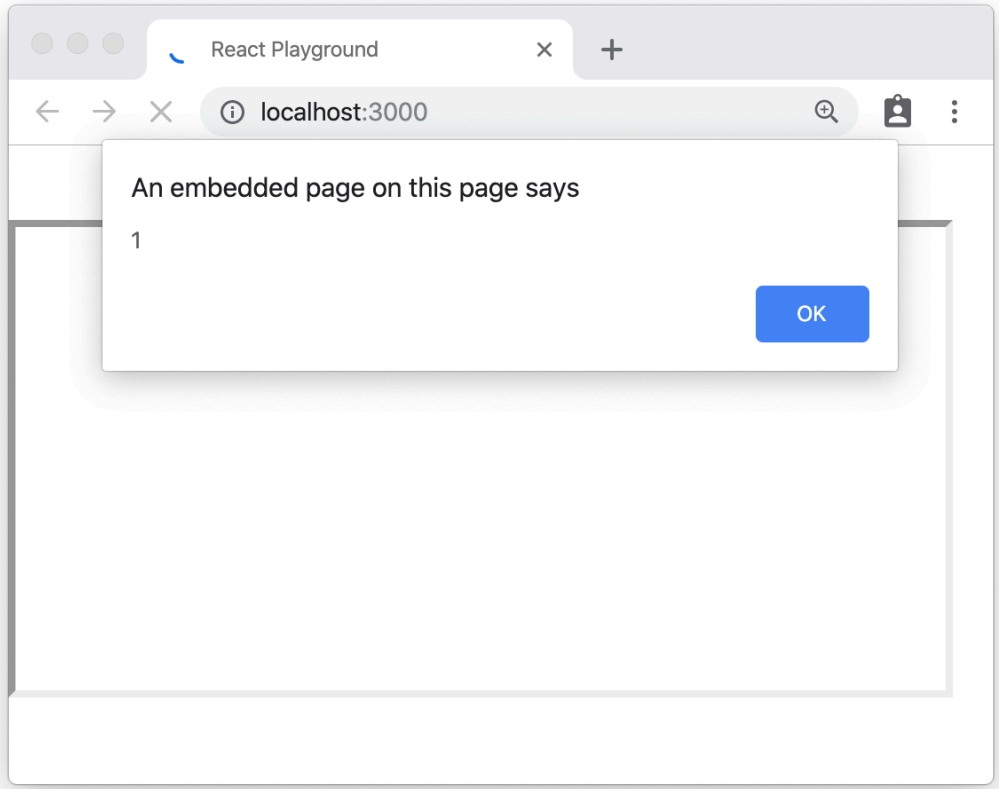
REACT CODE

```
return ( <div> { ReactHtmlParser(data) }</div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
<iframe src='data:text/html,<script>alert(1)</script>'>
```

RENDERED PAGE



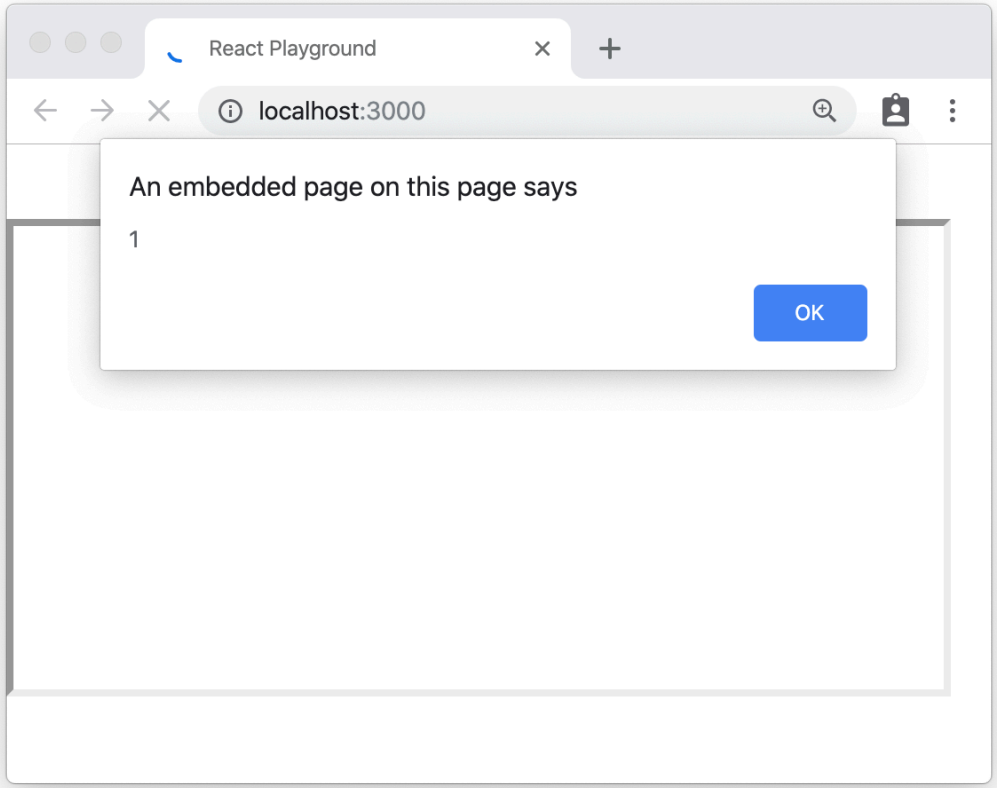
REACT CODE

```
return ( <div> { ReactHtmlParser(data) }</div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
<iframe src='data:text/html<svg onload=alert(1)>'>
```

RENDERED PAGE



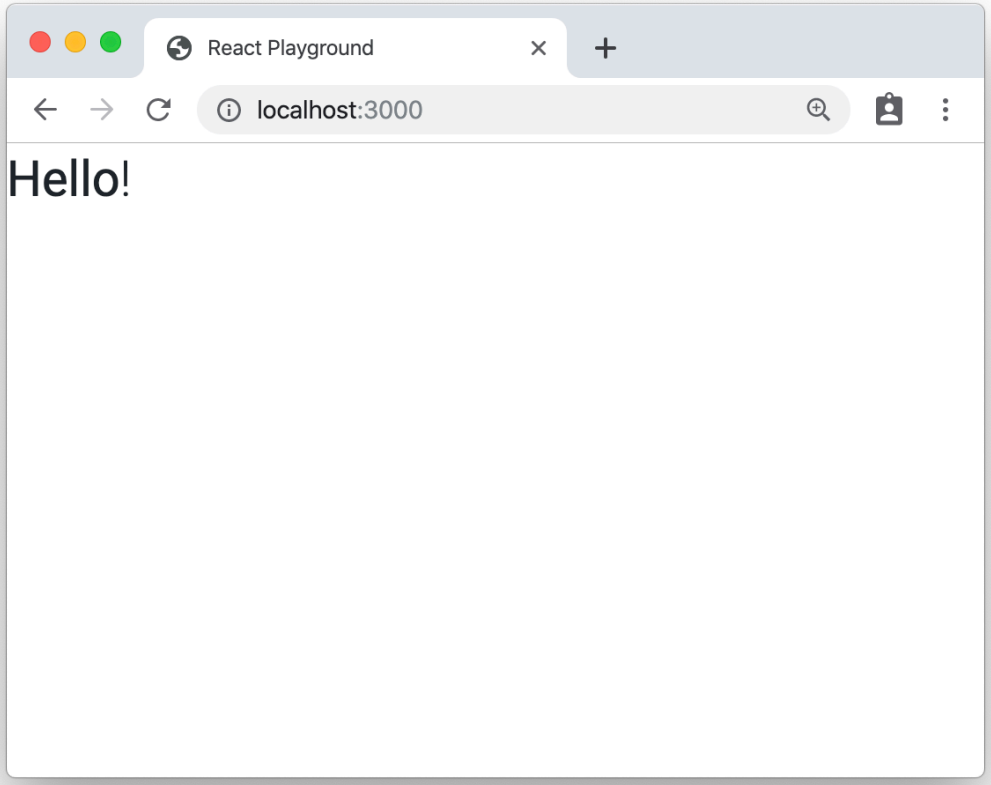
REACT CODE

```
return (
<div dangerouslySetInnerHTML={{__html:
DOMPurify.sanitize(data)}}></div> );
```

UNTRUSTED DATA

```
<b>Hello</b>!
<iframe src='data:text/html<svg onload=alert(1)>'>
```

RENDERED PAGE



Avoiding XSS in React is Still Hard

Ron Perris Follow
Apr 15, 2018 · 5 min read

```
const URL = require('url-parse')


function isSafe(url) {
  if (url.protocol === 'javascript:') return false


  return true
}


isSafe(URL(' javascript: alert(1)')) // Returns true
isSafe(URL('http://www.reactjs.org')) // Returns true
```

Avoiding XSS in React is Still Hard

Ron Perris Follow

Apr 15, 2018 · 5 min read

```
isSafe(dangerousURL, text) {
  const url = URL(dangerousURL, {})
  if (url.protocol === 'http:') return true
  if (url.protocol === 'https:') return true


  return false
}
```

Avoiding XSS in React

Use simple data binding in JSX

Only use dangerouslySetInnerHTML in combination with sanitization

Do not rely on parsers for security

```
class TestComponent extends React.Component {
    constructor(props) {
        super(props);
        this.placeholder = "Hello!";
        this.myComponent = React.createRef();
    }

    componentDidMount() {
        let realData = "...";
        this.myComponent.current.innerHTML = realData;
    }

    render() {
        return (
            <div ref={this.myComponent}>{this.placeholder}</div>
        )
    }
}
```

```
class TestComponent extends React.Component {
    constructor(props) {
        super(props);
        this.placeholder = "Hello!";
    }

    componentDidMount() {
        let realData = "...";
        ReactDOM.findDOMNode(this).innerHTML = realData;
    }

    render() {
        return (
            <div>{this.placeholder}</div>
        )
    }
}
```

findDOMNode

Search

- Repositories 3
- **Code 298K**
- Commits 8K
- Issues 13K
- Packages 0
- Marketplace 0
- Topics 0
- Wikis 86
- Users 0

Languages

JavaScript	177,126
JSX	54,111
TypeScript	11,565

298,782 code results

Sort: Best match

lngoquy1/NYTimes-TopStories-crossword

ref.txt

```
1  answer0000: ReactDOM.findDOMNode(this.refs.answer0000).value,
2  answer0001: ReactDOM.findDOMNode(this.refs.answer0001).value,
3  answer0002: ReactDOM.findDOMNode(this.refs.answer0002).value,
4  answer0003: ReactDOM.findDOMNode(this.refs.answer0003).value,
```

Text Showing the top three matches Last indexed on Jul 11, 2018

kkinnebrew/fiscality-ui

app/scripts/views/app/navigation.jsx

```
5    handleSummary: function() {
6
7      if (React.findDOMNode(this.refs.summary).classList.contains('selected'))
   return;
8
9      React.findDOMNode(this.refs.summary).classList.add('selected');
10     React.findDOMNode(this.refs.banking).classList.remove('selected');
```

JSX Showing the top two matches Last indexed on Jun 29, 2018

Avoiding XSS in React

Use simple data binding in JSX

Only use dangerouslySetInnerHTML in combination with sanitization

Do not rely on parsers for security

Do not put data in the DOM directly

So, we're f*cked?

Every developer should stick to secure coding guidelines

Build libraries to encapsulate dangerous behavior

Create only one accepted way to output safe HTML,
without every developer having to learn about DOMPurify

Use linting to find potentially dangerous code patterns

ESLint-plugin-React

React specific linting rules for ESLint

CROSS-SITE SCRIPTING (XSS)

React only applies automatic defenses for simple data binding

Watch out for React pitfalls when handling data

Use linting rules to scan your applications for dangerous patterns

> 97%

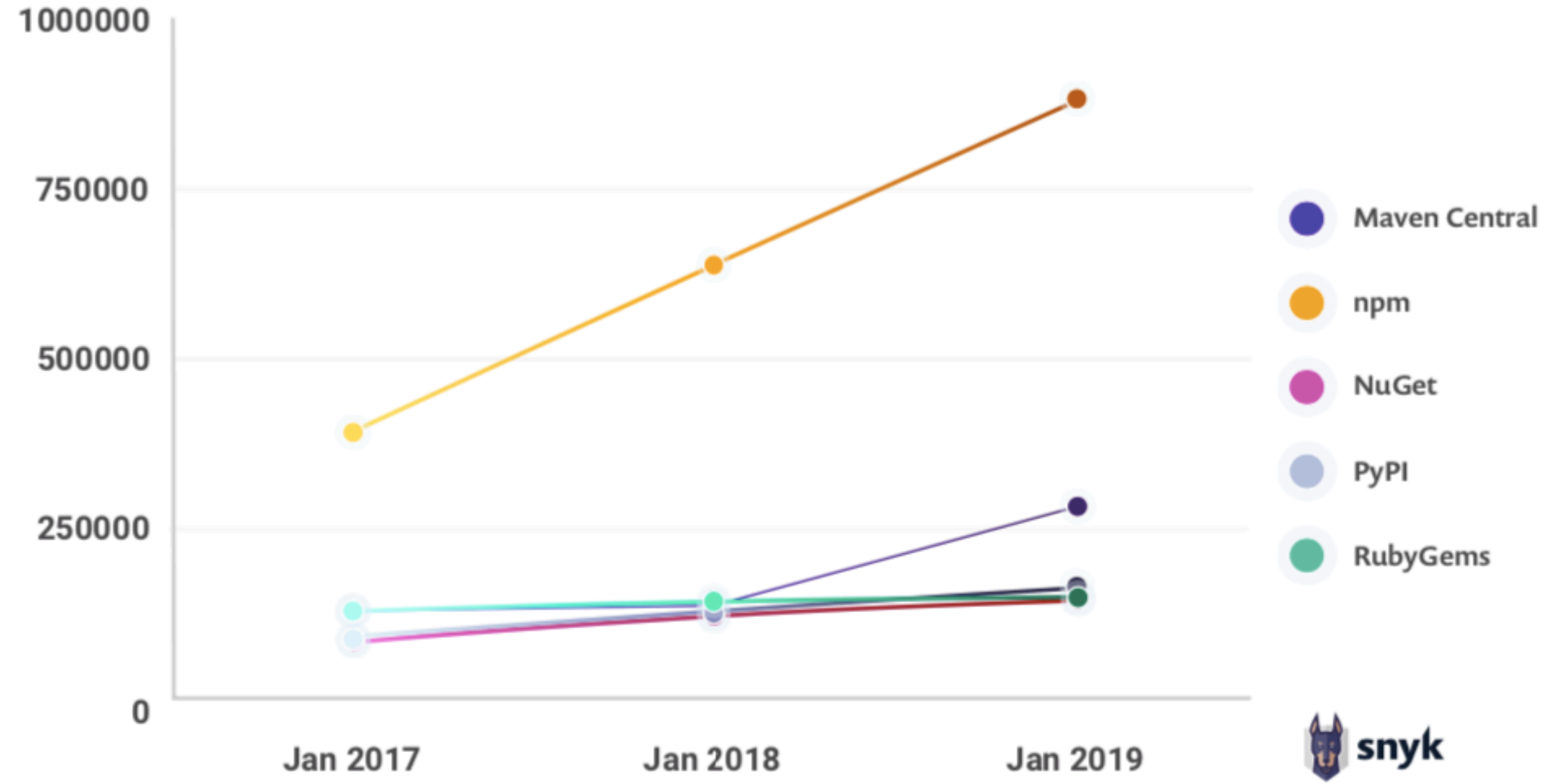
of code in a modern web app are dependencies

```
$ cat package.json
  ...
  "dependencies": {
    "@fortawesome/fontawesome-free": "^5.10.2",
    "bootstrap-css-only": "4.3.1",
    "cloc": "^2.5.0",
    "dompurify": "^2.0.7",
    "mdbreact": "4.19.1",
    "prop-types": "15.7.2",
    "react": "16.9.0",
    "react-dom": "16.9.0",
    "react-html-parser": "^2.0.2",
    "react-router-dom": "^5.0.1"
  },
  "devDependencies": {
    "react-scripts": "latest"
  },
  ...
```

```
$ cloc node_modules/
```

Language	files	blank	comment	code
JavaScript	17074	264815	271832	1482750
JSON	2199	624	0	238181
Markdown	1944	81253	4	205375
TypeScript	1844	6259	53636	49077
CSS	75	10156	478	38177
SVG	1594	0	49	28529
HTML	65	11671	24	19678
Sass	83	2207	465	17465
Perl	1	851	1323	11464
LESS	28	1010	101	9267
YAML	177	121	168	6302
C/C++ Header	21	1148	351	5984
JSX	10	389	353	2444
XML	8	233	11	2138
CoffeeScript	27	591	51	1513
Bourne Shell	10	204	140	991
Windows Module Definition	5	88	0	454
...				
SUM:	25194	381793	329060	2120568

Total packages indexed per ecosystem



the average npm module relies on

80 packages

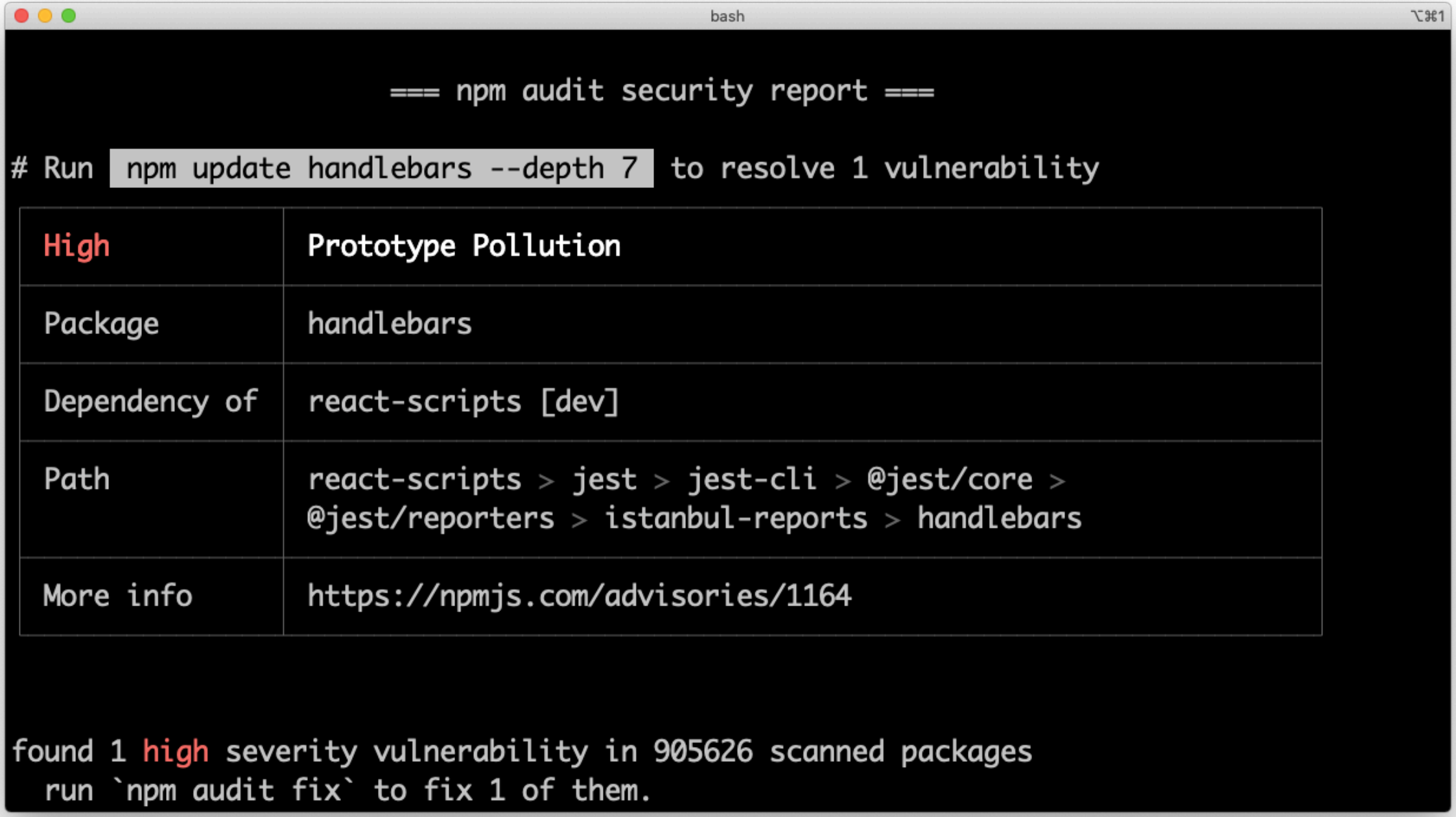
Small world with high risks: a study of security threats in the npm ecosystem

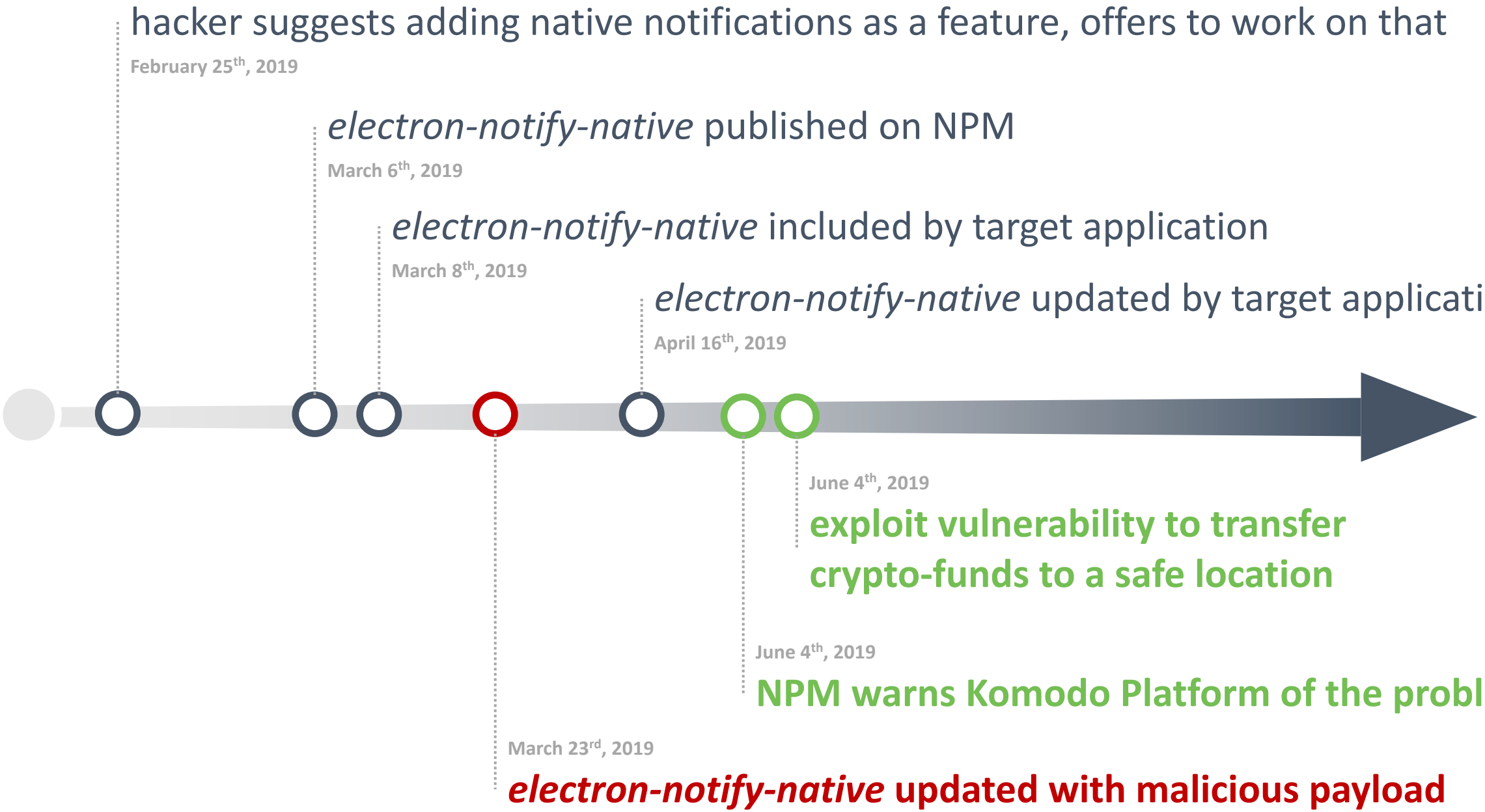
40%

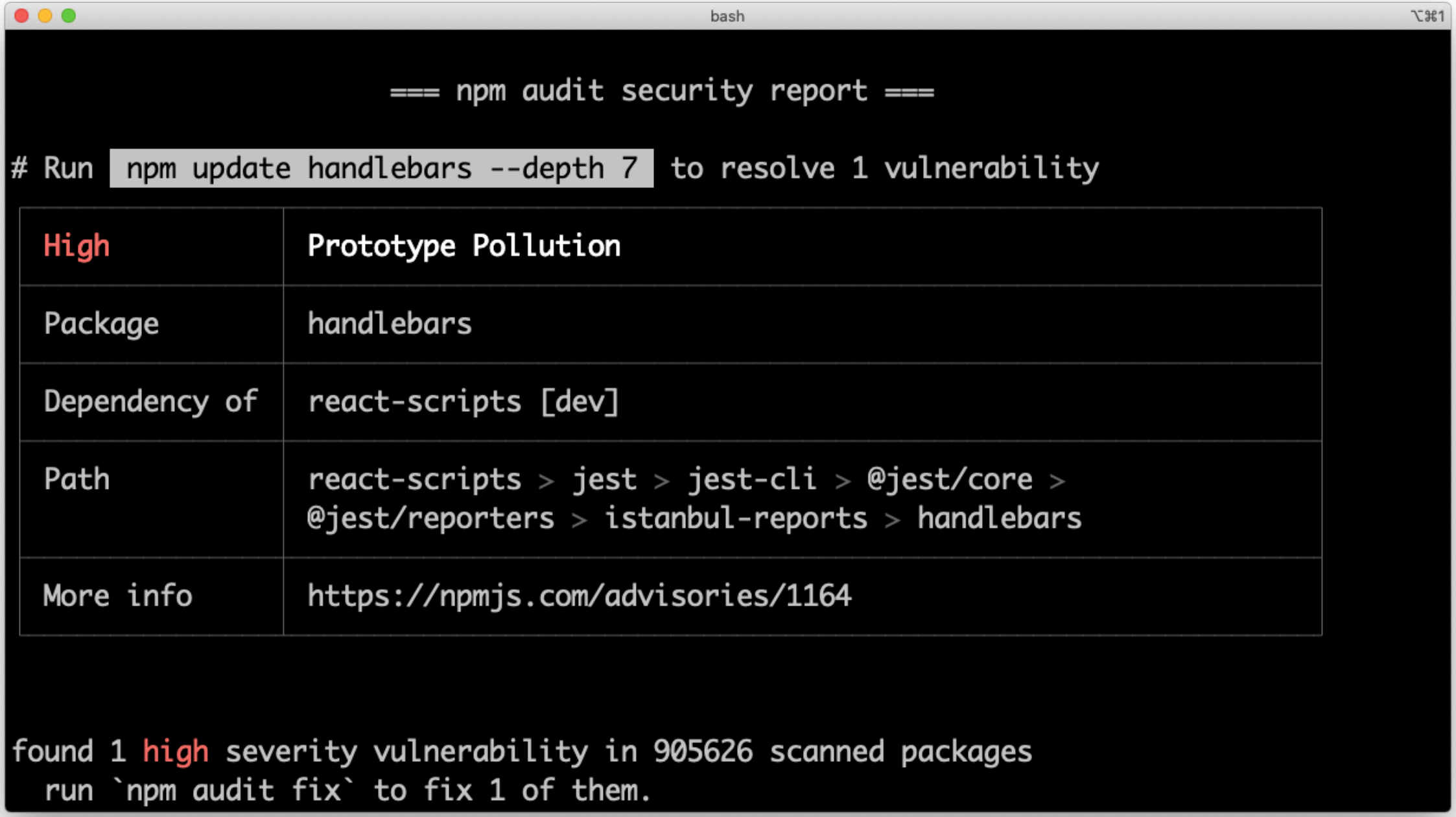
of packages rely on known vulnerable code*

*estimated by the authors of

Small world with high risks: a study of security threats in the npm ecosystem


bash
⌥⌘1
=== npm audit security report ===
Run npm update handlebars --depth 7 to resolve 1 vulnerability
High
Prototype Pollution
Package
handlebars
Dependency of
react-scripts [dev]
Path
react-scripts > jest > jest-cli > @jest/core > @jest/reporters > istanbul-reports > handlebars
More info
https://npmjs.com/advisories/1164
found 1 high severity vulnerability in 905626 scanned packages
run `npm audit fix` to fix 1 of them.


hacker suggests adding native notifications as a feature, offers to work on that
February 25th, 2019
electron-notify-native published on NPM
March 6th, 2019
electron-notify-native included by target application
March 8th, 2019
electron-notify-native updated by target application
April 16th, 2019
June 4th, 2019
exploit vulnerability to transfer crypto-funds to a safe location
June 4th, 2019
NPM warns Komodo Platform of the problem
March 23rd, 2019
electron-notify-native updated with malicious payload


bash
⌥⌘1
=== npm audit security report ===
Run npm update handlebars --depth 7 to resolve 1 vulnerability
High
Prototype Pollution
Package
handlebars
Dependency of
react-scripts [dev]
Path
react-scripts > jest > jest-cli > @jest/core > @jest/reporters > istanbul-reports > handlebars
More info
https://npmjs.com/advisories/1164
found 1 high severity vulnerability in 905626 scanned packages
run `npm audit fix` to fix 1 of them.

- Pulse
- Contributors
- Traffic
- Commits
- Code frequency
- **Dependency graph**
- Alerts
- Network
- Forks

Dependency graph

Dependencies | Dependents

We found potential security vulnerabilities in your dependencies.

Dependencies defined in these manifest files have known security vulnerabilities and should be updated:

restograde-angular/package-lock.json *5 vulnerabilities found*
reviewer-angular/package-lock.json *5 vulnerabilities found*

See security alerts

Only the owner of this repository can see this message.
Learn more about vulnerability alerts

These dependencies are defined in **pws-restograde**'s manifest files, such as **reviewer-angular/package-lock.json**, **reviewer-angular/package.json**, and **restograde-angular/package-lock.json**.

Snyk for Developers & DevOps

Snyk continuously monitors your application's dependencies and lets you quickly respond when new vulnerabilities are disclosed.

Fix your vulnerabilities

- **Single click fix - generate a fix PR from UI, CLI wizard**
- Upgrade - Automatically calculates the minimal direct dependency version upgrade needed
- Precision patch - Use patches backported by Snyk security team to fix when direct upgrade is not available or it'll take time to have upgrade implemented
- Automatic fix for new vulnerabilities - Automatically generate fix pull requests for newly discovered vulnerabilities

Equifax uses *Apache Struts 2* to build applications

a patched version of *Struts2* fixes a remote code execution vulnerability

March 7th, 2017

Equifax discovers the breach of their systems

July 29th, 2017

Equifax announces the breach

September 7th, 2017

May 2017

attackers escalate the attack to full-scale data exfiltration

March 10th, 2017

attackers start probing *Equifax* systems using the *Struts* vulnerability

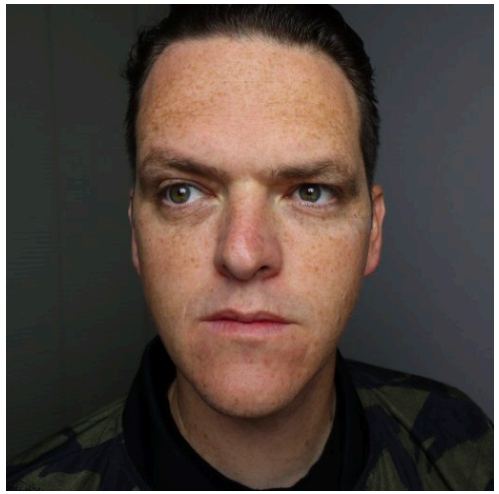
Securing your dependencies

Setup continuous monitoring for dependencies

Triage potential vulnerabilities for the urgency to patch

Keep applications up to date, to enable quick patching

Other people to listen to about React/JS security



Ron Perris

@ronperris

Lewis Ardern

@LewisArdern

Pragmatic Web Security

Security for developers

/in/PhilippeDeRyck

@PhilippeDeRyck

philippe@pragmaticwebsecurity.com