



SECURITY WITH NODE.JS

---

# BEST PRACTICES

David Petrasovic

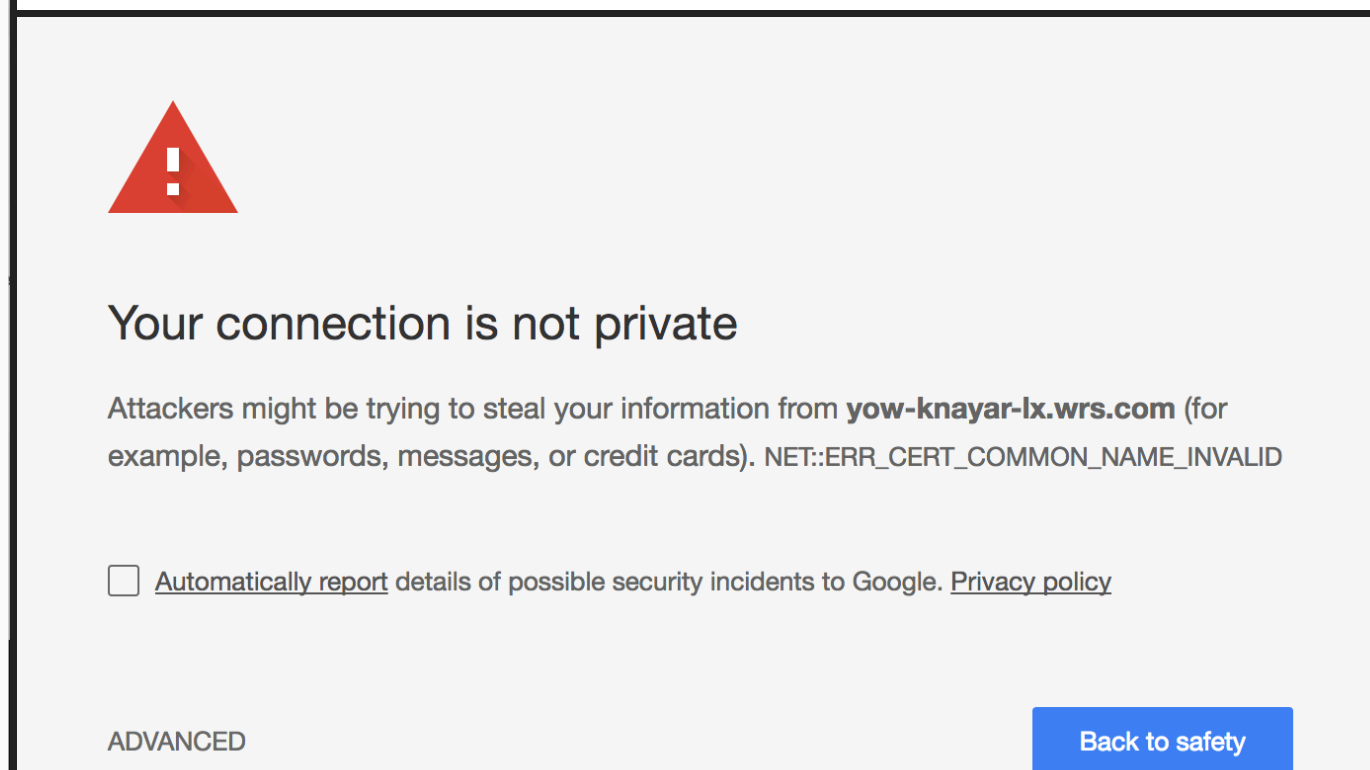
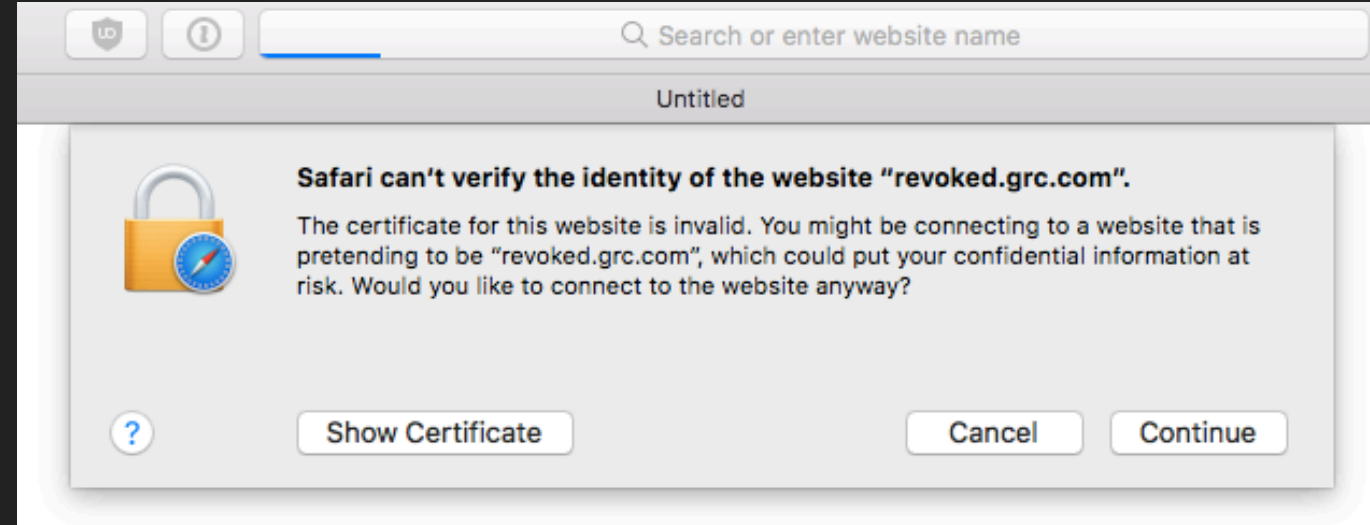
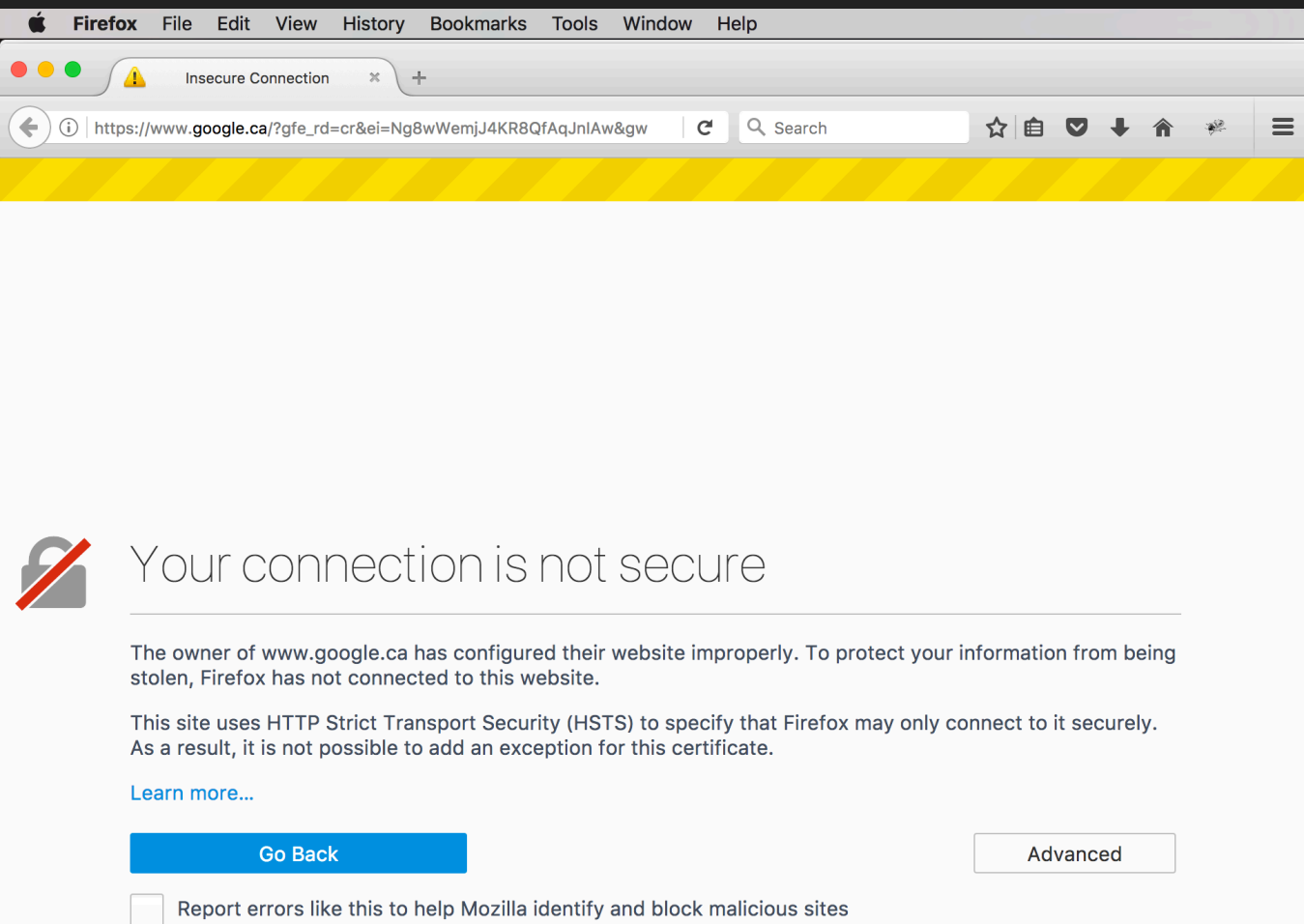
# USE TLS

- ▶ Transport Layer Security
- ▶ Let's Encrypt is FREE!
- ▶ Prevent Man In the Middle Attacks



<https://letsencrypt.org/>

# HOW TLS TELLS YOU SOMEONE IS WATCHING YOU (MITM)





---

USE HELMET



## USE HELMET

- ▶ Helmet sets HTTP headers for you
- ▶ Prevent Clickjacking, XSS, etc.

```
$ npm install --save helmet
```

```
var express = require('express')  
var helmet = require('helmet')  
  
var app = express()  
  
app.use(helmet())
```



HELMET

# CLICKJACK DEMO

## Congradulations! You won!!!

[Click Here for a free iPad!!!](#)

The screenshot shows the Chrome DevTools interface. The top bar includes navigation icons and tabs for Elements, Profiles, Network, Sources, Console, and Extensions. The Elements panel is active, showing the DOM tree. The selected element is an `iframe` with the following styles:

```

element.style {
}


iframe {
  height: 300px;
  width: 230px;
  position: absolute;
  top: -173px;
  left: -43px;
  opacity: 0;
}

iframe {
  border-width: 2px;
  border-style: inset;
  border-color: initial;
  border-image: initial;
}

```

The Console panel is also visible, showing the 'top' message and a 'Preserve log' checkbox. The bottom bar includes a filter input and buttons for All, Errors, Warnings, Info, Logs, and Debug.

# SCANNING DEPENDENCIES



**Yes. I always have coffee  
when I watch radar.  
You know that.**

# SCANNING DEPENDENCIES

## ▶ Node Security Platform (NSP)

```
$ npm i nsp -g
$ nsp check
```

```
YOW-DPETRASO-M:hdc-ui dave$ npm i nsp -g
/usr/local/bin/nsp -> /usr/local/lib/node_modules/nsp/bin/nsp
nsp@2.6.3 /usr/local/lib/node_modules/nsp
├─ path-is-absolute@1.0.0
├─ nodesecurity-npm-utils@5.0.0
├─ cvss@1.0.1
├─ semver@5.1.0
├─ chalk@1.1.3 (ansi-styles@2.2.1, escape-string-regexp@1.0.5, supports-color@2.0.0, has
├─ rc@1.1.6 (ini@1.3.4, deep-extend@0.4.1, strip-json-comments@1.0.4, minimist@1.2.0)
├─ cli-table@0.3.1 (colors@1.0.3)
├─ wreck@6.3.0 (boom@2.10.1, hoek@2.16.3)
├─ https-proxy-agent@1.0.0 (extend@3.0.0, agent-base@2.0.1, debug@2.2.0)
├─ subcommand@2.0.3 (cliclopts@1.1.1, xtend@4.0.1, minimist@1.2.0, debug@2.2.0)
├─ joi@6.10.1 (topo@1.1.0, isemail@1.2.0, hoek@2.16.3, moment@2.12.0)
YOW-DPETRASO-M:hdc-ui dave$ nsp check
(+) 23 vulnerabilities found
```

	DoS due to excessively large websocket message
Name	ws
CVSS	7.5 (High)
Installed	0.8.0
Vulnerable	<=1.1.0
Patched	>=1.1.1
Path	uiapp@1.0.0 > socket.io@1.3.7 > engine.io@1.5.4 > ws@0.8.0
More Info	<a href="https://nodesecurity.io/advisories/120">https://nodesecurity.io/advisories/120</a>

	Denial-of-Service Memory Exhaustion
Name	qs
CVSS	7.5 (High)
Installed	0.6.6
Vulnerable	<1.0.0
Patched	>= 1.x
Path	uiapp@1.0.0 > express@4.0.0 > qs@0.6.6
More Info	<a href="https://nodesecurity.io/advisories/29">https://nodesecurity.io/advisories/29</a>

	Regular Expression Denial of Service
Name	negotiator
CVSS	7.5 (High)
Installed	0.4.7
Vulnerable	<= 0.6.0
Patched	>= 0.6.1
Path	uiapp@1.0.0 > express@4.0.0 > negotiator@0.4.7 > negotiator@0.6.0



## SCANNING DEPENDENCIES

- ▶ Using Snyk (So Now You Know)

```
$ npm install -g snyk
```

- ▶ Create Account

```
$ snyk auth
```

```
$ snyk test
```

- ▶ Guided wizard to update dependencies

```
$ snyk wizard
```

- ▶ Free for open source



## REDOS

- ▶ ReDoS: Regular Expression Denial of Service

- ▶ Example of bad regex: `([a-zA-z]+)* (a+)+`

- ▶ Can hang with input like: `aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa!`

- ▶ Use safe-regex to test

```
$ npm install safe-regex
```

```
$ node safe.js '(foo|bar)*'  
true
```

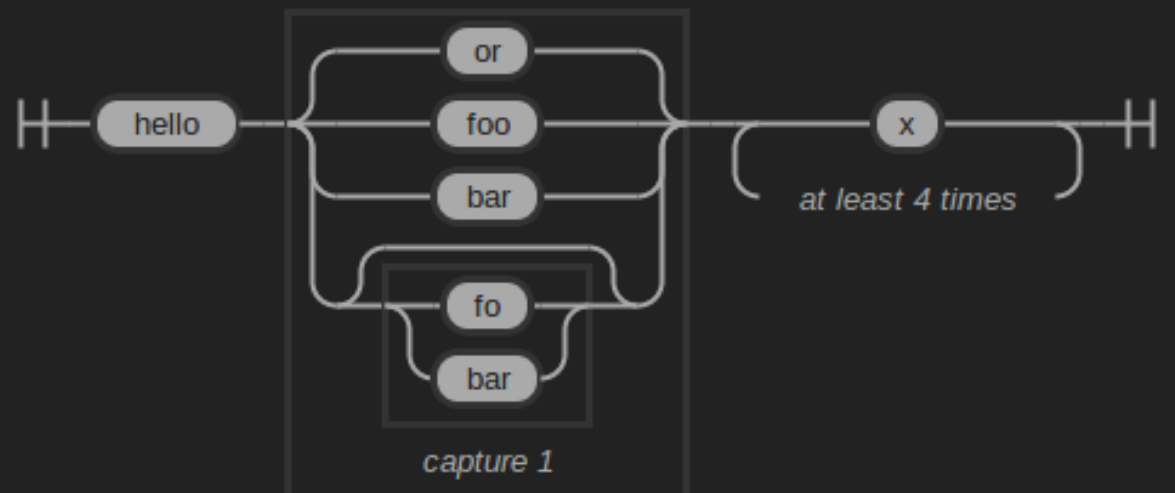
```
$ node safe.js '(a+){10}'  
false
```

## WRITING A REGEX

- ▶ regex-railroad-diagram package for Atom

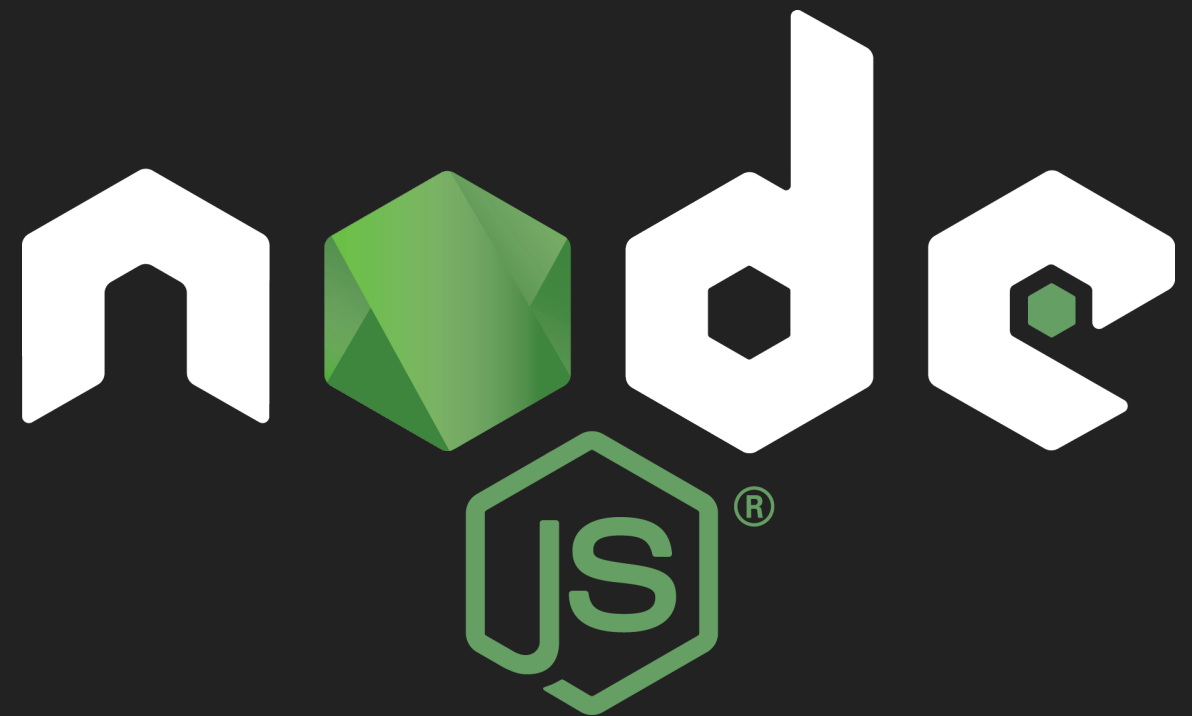


```
railroad-diagrams.js x regex-railroad-diagram-view.coffee o regex-to-railc
51 ..serialize: ->~
52 ~
53 ..# Tear down any state and detach~
54 ..destroy: ->~
55 ....@detach()~
56 ~
57 ..getRegexScope: (scope) ->~
58 ....scopeName = []~
59 ....for name in scope~
60 .....scopeName.push name~
61 ~
62 .....if /^string\.regexp/.test name~
63 .....scopeName~
64 ~
65 ....x = /hello(or|foo|bar|(?fo|bar)?)x{4,}/~
66 ~
67 ....false~
68 ~
69 ~
70 ..showRailRoadDiagram: (regex) ->~
71 ....rr = atom.workspaceView.find '.regex-railroad-diagram
72 ....if not rr.length~
73 .....# create current diff~
74 .....@hide()~
75 ~
76 .....# append to "ones"
```

A railroad diagram for the regular expression `/hello(or|foo|bar|(?fo|bar)?)x{4,}/`. The diagram starts with a start symbol (two vertical bars), followed by a terminal node labeled 'hello'. This is followed by a large bracketed group containing four parallel paths: 'or', 'foo', 'bar', and a 'capture 1' group. The 'capture 1' group is a smaller bracketed box containing 'fo' and 'bar' in parallel. After the large group, there is a terminal node labeled 'x' followed by a bracket labeled 'at least 4 times', and finally an end symbol (two vertical bars).

## FINAL THOUGHTS

- ▶ Keep Node.js up to date
- ▶ Careful with `exec()`
- ▶ Do not reinvent the wheel
  - ▶ passport (passportjs.org)  
`$ npm install passport`
  - ▶ node-postgress  
`$ npm install pg`
  - ▶ etc.





**QUESTIONS?**

**END**