CSE 127: Computer Security

Modern client-side defenses

Deian Stefan

Today

How can we build **flexible** and **secure** client-side web applications (from vulnerable/untrusted components)

Modern web sites are complicated

Many acting parties on a site

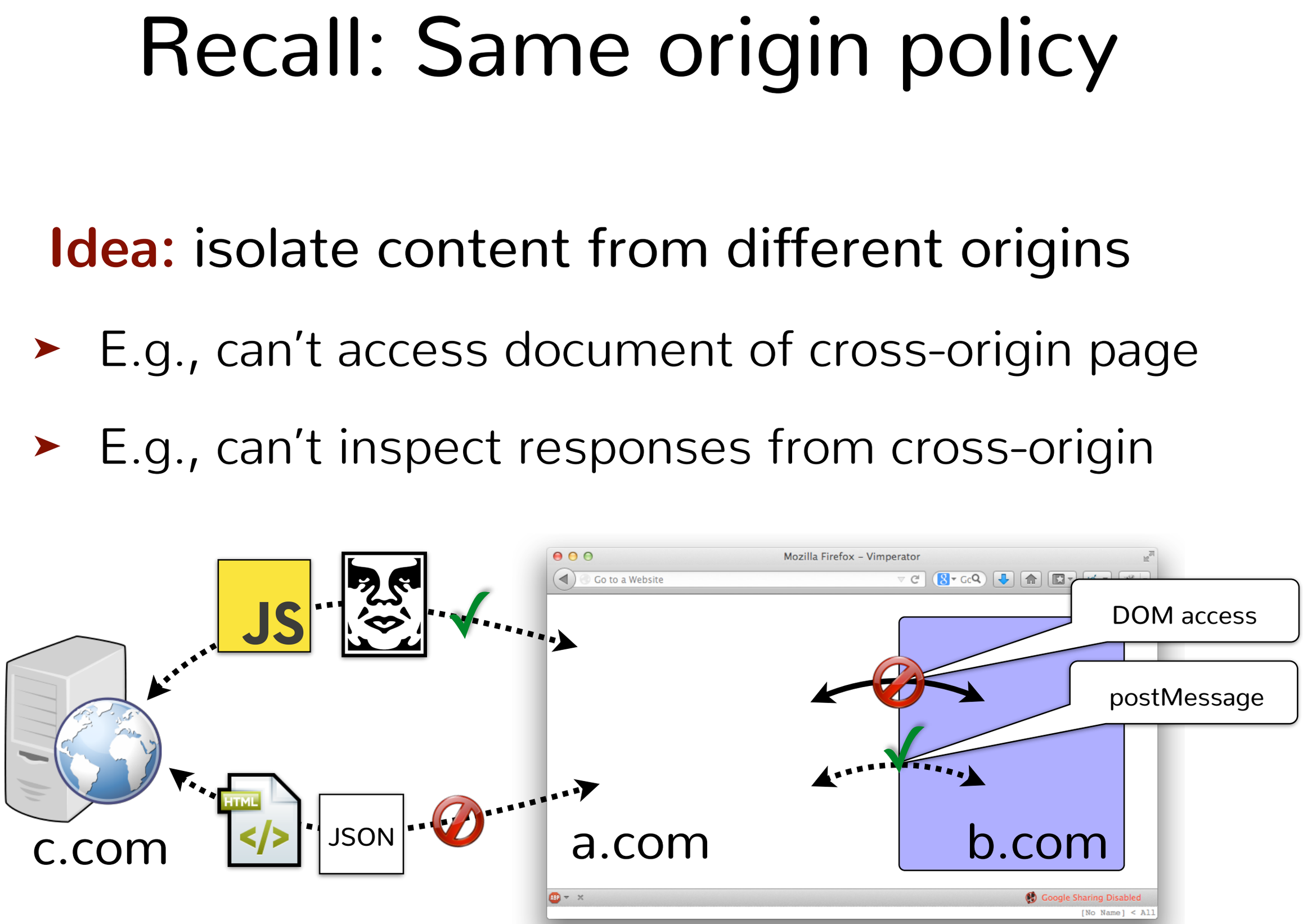
- Page developer
- Library developers
- Service providers
- Data provides
- Ad providers
- CDNs
- Network provider

- How do we protect page from ads/services?
- How to share data with a cross-origin page?
- How to protect one user from another's content?
- How do we protect the page from a library?
- How do we protect the page from the CDN?
- How do we protect the page from network provider?

Recall: Same origin policy

Idea: isolate content from different origins

- E.g., can't access document of cross-origin page
- E.g., can't inspect responses from cross-origin



Why is the SOP not good enough?

The SOP is not strict enough

- Third-party libs run with privilege of the page
- Code within page can arbitrarily leak/corrupt data
  - How? So, how should we isolate untrusted code?

The SOP is not strict enough

- Third-party libs run with privilege of the page
- Code within page can arbitrarily leak/corrupt data
  - How? So, how should we isolate untrusted code?
- iframes isolation is limited
  - Can't isolate user-provided content from page (why?)
  - Can't isolate third-party ad placed in iframe (why?)

The New York Times

September 15, 2009

The SOP is not flexible enough

- Can't read cross-origin responses
  - What if we want to fetch data from provider.com?
  - JSON with padding (JSONP)
    - To fetch data, insert new script tag:
      `<script src="https://provider.com/getData?cb=f"></script>`
    - To share data, reply back with script wrapping data
      `f({ ...data... })`

http://example.com

provider.com

Why is JSONP is a terrible thing?

Why is JSONP is a terrible thing?

- Provider data can easily be leaked (CSRF)
- Page is not protected from provider (XSS)

The SOP is also not enough security-wise

Outline: modern mechanisms

- iframe sandbox
- Content security policy (CSP)
- HTTP strict transport security (HSTS)
- Subresource integrity (SRI)
- Cross-origin resource sharing (CORS)

iframe sandbox

Idea: restrict actions iframe can perform

Approach: set sandbox attribute, by default:

- disallows JavaScript and triggers (autofocus, autoplay videos etc.)
- disallows form submission
- disallows popups
- disallows navigating embedding page
- runs page in unique origin: no storage/cookies

Grant privileges explicitly

Can enable dangerous features with:

- **allow-scripts:** allows JS + triggers (e.g., autofocus)
- **allow-forms:** allow form submission
- **allow-popups:** allow iframe to create popups
- **allow-top-navigation:** allow breaking out of frame
- **allow-same-origin:** retain original origin

Grant privileges explicitly

The **`sandbox`** attribute, when specified, enables a set of extra restrictions on any content hosted by the `iframe`. Its value must be an unordered set of unique space-separated tokens that are ASCII case-insensitive. The allowed values are `allow-forms`, `allow-modals`, `allow-orientation-lock`, `allow-pointer-lock`, `allow-popups`, `allow-popups-to-escape-sandbox`, `allow-presentation`, `allow-same-origin`, `allow-scripts`, `allow-top-navigation`, `allow-top-navigation-by-user-activation`, and `allow-downloads`.

When the attribute is set, the content is treated as being from a unique origin, forms, scripts, and various potentially annoying APIs are disabled, links are prevented from targeting other browsing contexts, and plugins are secured. The `allow-same-origin` keyword causes the content to be treated as being from its real origin instead of forcing it into a unique origin; the `allow-top-navigation` keyword allows the content to navigate its top-level browsing context; the `allow-top-navigation-by-user-activation` keyword behaves similarly but allows such navigation only when the browsing context's active window has transient activation; and the `allow-forms`, `allow-modals`, `allow-orientation-lock`, `allow-pointer-lock`, `allow-popups`, `allow-presentation`, `allow-scripts`, and `allow-popups-to-escape-sandbox` keywords re-enable forms, modal dialogs, screen orientation lock, the pointer lock API, popups, the presentation API, scripts, and the creation of unsandboxed auxiliary browsing contexts respectively. [POINTERLOCK] [SCREENORIENTATION] [PRESENTATION]

The `allow-top-navigation` and `allow-top-navigation-by-user-activation` keywords must not both be specified, as doing so is redundant; only `allow-top-navigation` will have an effect in such non-conformant markup.

⚠Warning!

Setting both the `allow-scripts` and `allow-same-origin` keywords together when the embedded page has the same origin as the page containing the `iframe` allows the embedded page to simply remove the `sandbox` attribute and then reload itself, effectively breaking out of the sandbox altogether.

What can you do with iframe sandbox?

- Run content in iframe with least privilege
  - Only grant content privileges it needs
- Privilege separate page into multiple iframes
  - Split different parts of page into sandboxed iframes

Least privilege: twitter button

Least privilege: twitter button

- Let's embed button on our site
  - How do they suggest doing it?

```
<a href="https://twitter.com/share?ref_src=twsrc%5Etfw"
class="twitter-share-button" data-show-count="false">Tweet</
a><script async src="https://platform.twitter.com/widgets.js"
charset="utf-8"></script>
```

  - What's the problem with this embedding approach?

Least privilege: twitter button

https://platform.twitter.com/widgets.js


```
Function&&Function.prototype&&Function.prototype.bind&&(/(MSIE ([6789]|10|11))|Trident/.test(navigator.userAgent)||
(window.__twttr&&window.__twttr.widgets&&window.__twttr.widgets.loaded&&window.twttr.widgets.load&&window.twttr.widgets.load()
,window.__twttr&&window.__twttr.widgets&&window.__twttr.widgets.init||function(t){function e(e){for(var
n,i,o=e[0],s=e[1],a=0,c=[];a<o.length;a++)i=o[a],r[i]&&c.push(r[i][0]),r[i]=0;for(n in
s)Object.prototype.hasOwnProperty.call(s,n)&&(t[n]=s[n]);for(u&&u(e);c.length;)c.shift()()}var n={},r={1:0};function
i(e){if(n[e])return n[e].exports;var r=n[e]={i:e,l:!1,exports:{}};return
t[e].call(r.exports,r,r.exports,i),r.l=!0,r.exports}i.e=function(t){var e=[],n=r[t];if(0!==n)if(n)e.push(n[2]);else{var o=new
Promise(function(e,i){n=r[t]=[e,i]});e.push(n[2]=o);var s,a=document.getElementsByTagName("head")
[0],u=document.createElement("script");u.charset="utf-8",u.timeout=120,i.nc&&u.setAttribute("nonce",i.nc),u.src=function(t)
{return i.p+"js/"+
({0:"moment~timeline~tweet",2:"dm_button",3:"button",4:"moment",5:"periscope_on_air",6:"timeline",7:"horizon_tweet",8:"tweet"}
[t]||t)+"."+
{0:"ae149926685a43cb146e35371430188e",2:"fe973228b5a3f65e3d8a0dd10b553d17",3:"63c51c903061d0dbd843c41e8a00aa5a",4:"50cf823c7bf
26ac484f84f086ebc4bff",5:"97b8e8f03584fdf6deeb9dfd0bf9b2d5",6:"687eed636a16648c9f0b1f72d7fa68bd",7:"716ef7f4c155526f8ec8e60dbd
2fbf56",8:"e71d9c824fc78e1a4e43d645b66bf574"}[t]+".js"}(t),s=function(e){u.onerror=u.onload=null,clearTimeout(c);var n=r[t];
if(0!==n){if(n){var i=e&&("load"===e.type?"missing":e.type),o=e&&e.target&&e.target.src,s=new Error("Loading chunk "+t+"
failed.\n("+i+": "+o+")");s.type=i,s.request=o,n[1](s)}r[t]=void 0}};var c=setTimeout(function()
{s({type:"timeout",target:u})},12e4);u.onerror=u.onload=s,a.appendChild(u)}return
Promise.all(e)},i.m=t,i.c=n,i.d=function(t,e,n){i.o(t,e)||Object.defineProperty(t,e,{enumerable:!0,get:n})},i.r=function(t)
{"undefined"!=typeof Symbol&&Symbol.toStringTag&&Object.defineProperty(t,Symbol.toStringTag,
{value:"Module"}),Object.defineProperty(t,"__esModule",{value:!0})},i.t=function(t,e){if(1&e&&(t=i(t)),8&e)return
t;if(4&e&&"object"==typeof t&&t&&t.__esModule)return t;var n=Object.create(null);if(i.r(n),Object.defineProperty(n,"default",
{enumerable:!0,value:t}),2&e&&"string"!=typeof t)for(var r in t)i.d(n,r,function(e){return t[e]}.bind(null,r));return
n},i.n=function(t){var e=t&&t.__esModule?function(){return t.default}:function(){return t};return
i.d(e,"a",e),e},i.o=function(t,e){return Object.prototype.hasOwnProperty.call(t,e)},i.p="https://platform.twitter.com
/",i.oe=function(t){throw console.error(t),t};var o=window.__twttrll=window.__twttrll||[],s=o.push.bind(o);
o.push=e,o=o.slice();for(var a=0;a<o.length;a++)e(o[a]);var u=s;i(i.s=92)}([function(t,e,n){var r=n(1);function i(t,e){var
n;for(n in t)t.hasOwnProperty&&!t.hasOwnProperty(n)||e(n,t[n]);return t}function o(t){return{}.toString.call(t).match(/\s([a-
zA-Z]+)/)[1].toLowerCase()}function s(t){return t===Object(t)}function a(t){var
e;if(!s(t))return!1;if(Object.keys)return!Object.keys(t).length;for(e in t)if(t.hasOwnProperty(e))return!1;return!0}function
u(t){return t?Array.prototype.slice.call(t):[]}t.exports={aug:function(t){return u(arguments).slice(1).forEach(function(e)
{i(e,function(e,n){t[e]=n})}),t},async:function(t,e){r.setTimeout(function(){t.call(e||null)},0)},compact:function t(e){return
i(e,function(n,r){s(r)&&(t(r),a(r)&&delete e[n]),void 0!==r&&null!==r&&""!==r||delete e[n]}),e},contains:function(t,e){return!
(!t||!t.indexOf)&&t.indexOf(e)>-1},forIn:i,isObject:s,isEmptyObject:a,toType:o,isType:function(t,e){return
t==o(e)},toRealArray:u}},function(t,e,n){t.exports=window},function(t,e,n){var r=n(6);t.exports=function(){var t=this;
this.promise=new r(function(e,n){t.resolve=e,t.reject=n})}},function(t,e,n){var r=n(11),i=/(?:^|(?:https?:)?\/\/(?:www
\.)?twitter\.com(?::\d+)?(?:\/intent\/(?:follow|user)\/?\?screen_name=|(?:\/#!)?\/))@?([\w]+)(?:\?|&|$)/i,o=/(?:^|(?:https?:)?
\/\/(?:www\.)?twitter\.com(?::\d+)?\/(?:#!\/)?[\w_]+\/status(?:es)?\/)(\d+)/i,s=/^http(s?):\/\/(\w+\.)*twitter\.com([:
/]|$)/i,a=/^http(s?):\/\/(ton|pbs)\.twimg\.com/,u=/^#?([^.,<>!\s/#\-()'"]+)$/,c=/twitter\.com(?::\d{2,4})?\/intent\/(\w+)
/,d=/^https?:\/\/(?:www\.)?twitter\.com\/\w+\/timelines\/(\d+)/i,f=/^https?:\/\/(?:www\.)?twitter\.com\/i\/moments\/(\d+)
/i,l=/^https?:\/\/(?:www\.)?twitter\.com\/(\w+)\/(?:likes|favorites)/i,h=/^https?:\/\/(?:www\.)?twitter\.com\/(\w+)\/lists
\/([\w-%]+)/i,p=/^https?:\/\/(?:www\.)?twitter\.com\/i\/live\/(\d+)/i,m=/^https?:\/\/syndication\.twitter\.com\/settings
```

Least privilege: twitter button

- Better approach: put it in an iframe iframe

```
<iframe src="https://platform.twitter.com/widgets/tweet_button.html"
   style="border: 0; width:130px; height:20px;"></iframe>
```

- Is this good enough?
- What can a malicious/compromised twitter still do?

Least privilege: twitter button

- Use iframe sandbox: remove all permissions and then enable JavaScript, popups, etc.

```
<iframe src="https://platform.twitter.com/widgets/tweet_button.html"
   sandbox="allow-same-origin allow-scripts allow-popups allow-forms"
   style="border: 0; width:130px; height:20px;"></iframe>
```

- Is the allow-same-origin unsafe?
- Why do you think we need all the other bits?

Privilege separation: blog feed

- Typically include user content inline:

```
<div class="post">
  <div class="author">{{post.author}}</div>
  <div class="body">{{post.body}}</div>
</div>
```

  - Problem with this?

Privilege separation: blog feed

- Typically include user content inline:

```
<div class="post">
  <div class="author">{{post.author}}</div>
  <div class="body">{{post.body}}</div>
</div>
```

  - Problem with this?

- With iframe sandbox:

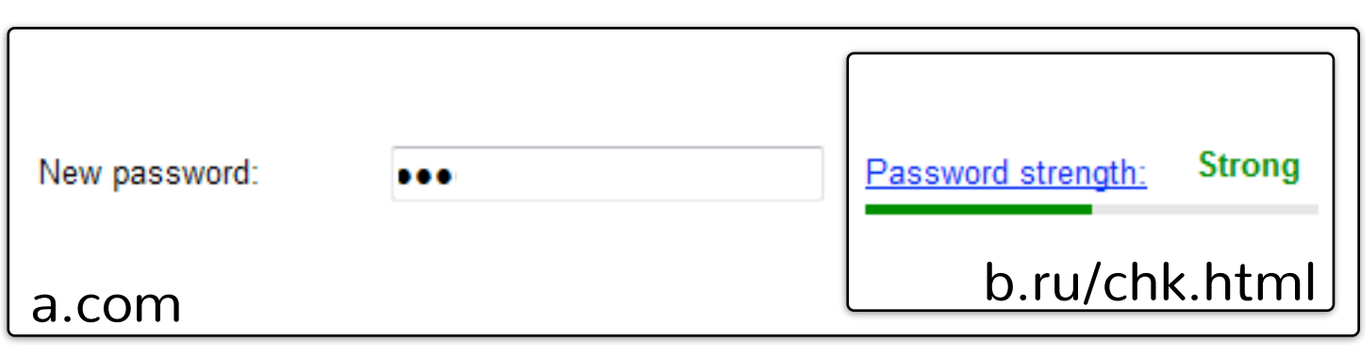
```
<iframe sandbox srcdoc="...
<div class="post">
  <div class="author">{{post.author}}</div>
  <div class="body">{{post.body}}</div>
</div>..."></iframe>
```

  - May need allow-scripts - why?
  - Is allow-same-origin safe too?

What are some limitations of iframe sandbox?

Too strict vs. not strict enough

- Consider running library in sandboxed iframes
  - E.g., password strength checker



  - **Desired guarantee:** checker cannot leak password
- Problem: sandbox does not restrict exfiltration
  - Can use XHR to write password to b.ru

Too strict vs. not strict enough

- Can we limit the origins that the page (iframe or otherwise) can talk talk to?
  - Can only leak to a trusted set of origins
  - Gives us a more fine-grained notion of least privilege
- This can also prevent or limit damages due to XSS

Outline: modern mechanisms

- iframe sandbox (quick refresher)
- Content security policy (CSP)
- HTTP strict transport security (HSTS)
- Subresource integrity (SRI)
- Cross-origin resource sharing (CORS)

Content Security Policy (CSP)

- Idea: restrict resource loading to a whitelist
  - By restricting to whom page can talk to: restrict where data is leaked!

- Approach: send page with CSP header that contains fine-grained directives
  - E.g., allow loads from CDN, no frames, no plugins

```
Content-Security-Policy: default-src https://cdn.example.net;
                         child-src 'none'; object-src 'none'
```

script-src: where you can load scripts from

connect-src: limits the origins you can XHR to

font-src: where to fetch web fonts form

form-action: where forms can be submitted

child-src: where to load frames/workers from

img-src: where to load images from

...

default-src: default fallback

Special keywords

- ‘none’ - match nothing
- ‘self’ - match this origin
- ‘unsafe-inline’ - allow unsafe JS & CSS
- ‘unsafe-eval’ - allow unsafe eval (and the like)
- http: - match anything with http scheme
- https: - match anything with https scheme
- * - match anything

How can CSP help with XSS?

- If you list all places you can load scripts from:
  - Only execute code from trusted origins
  - Remaining vector for attack: inline scripts
- CSP by default disallows inline scripts
  - If scripts are enabled at least it disallows eval

Adoption challenge

- Problem: inline scripts are widely-used
  - Page authors use the ‘unsafe-inline' directive
  - Is this a problem?

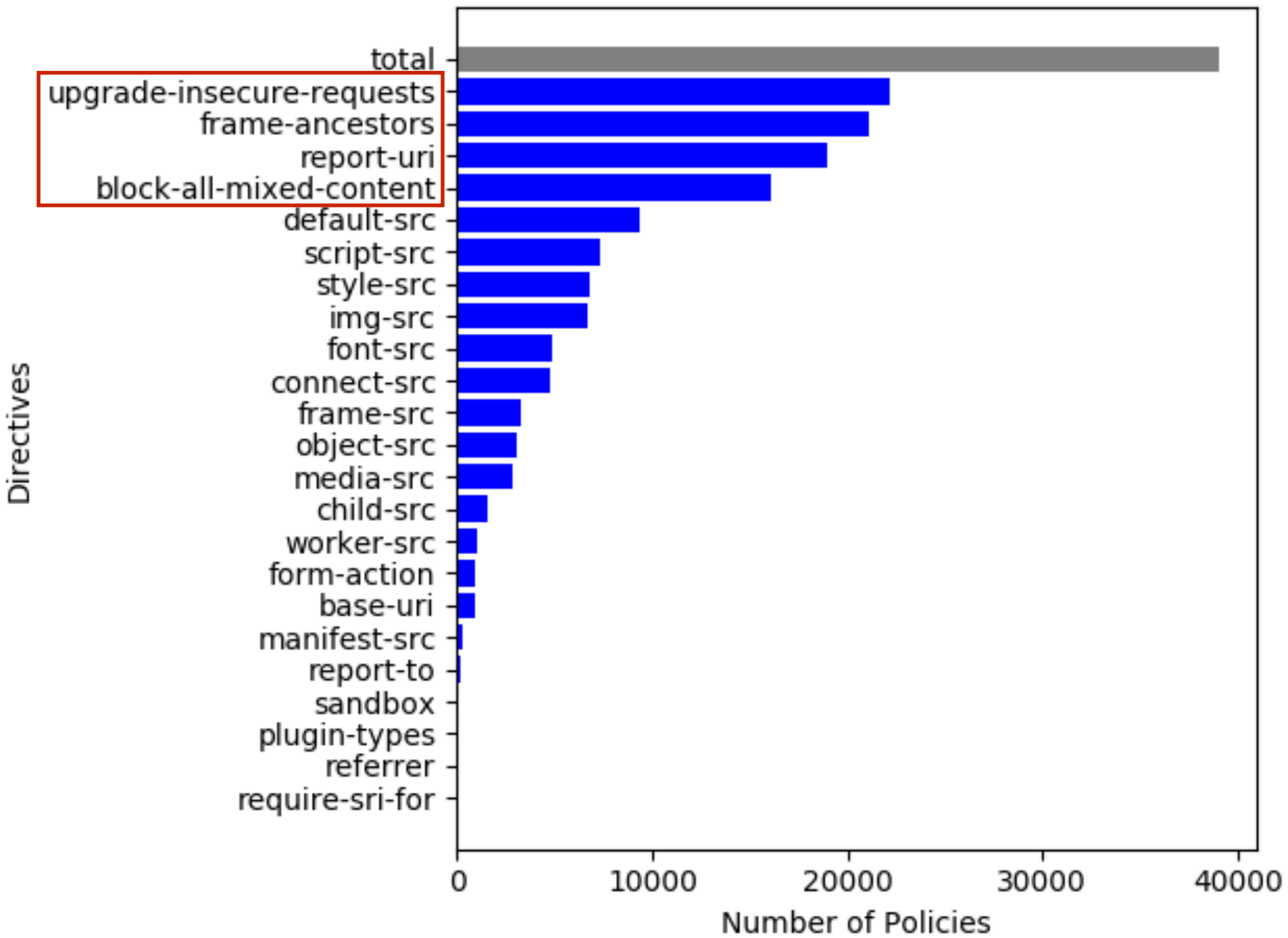
Adoption challenge

- Problem: inline scripts are widely-used
  - Page authors use the ‘unsafe-inline' directive
  - Is this a problem?
- Solution: ~~script nonce~~ and script hash
  - Allow scripts that have a particular hash
  - Allow scripts that have a specific nonce

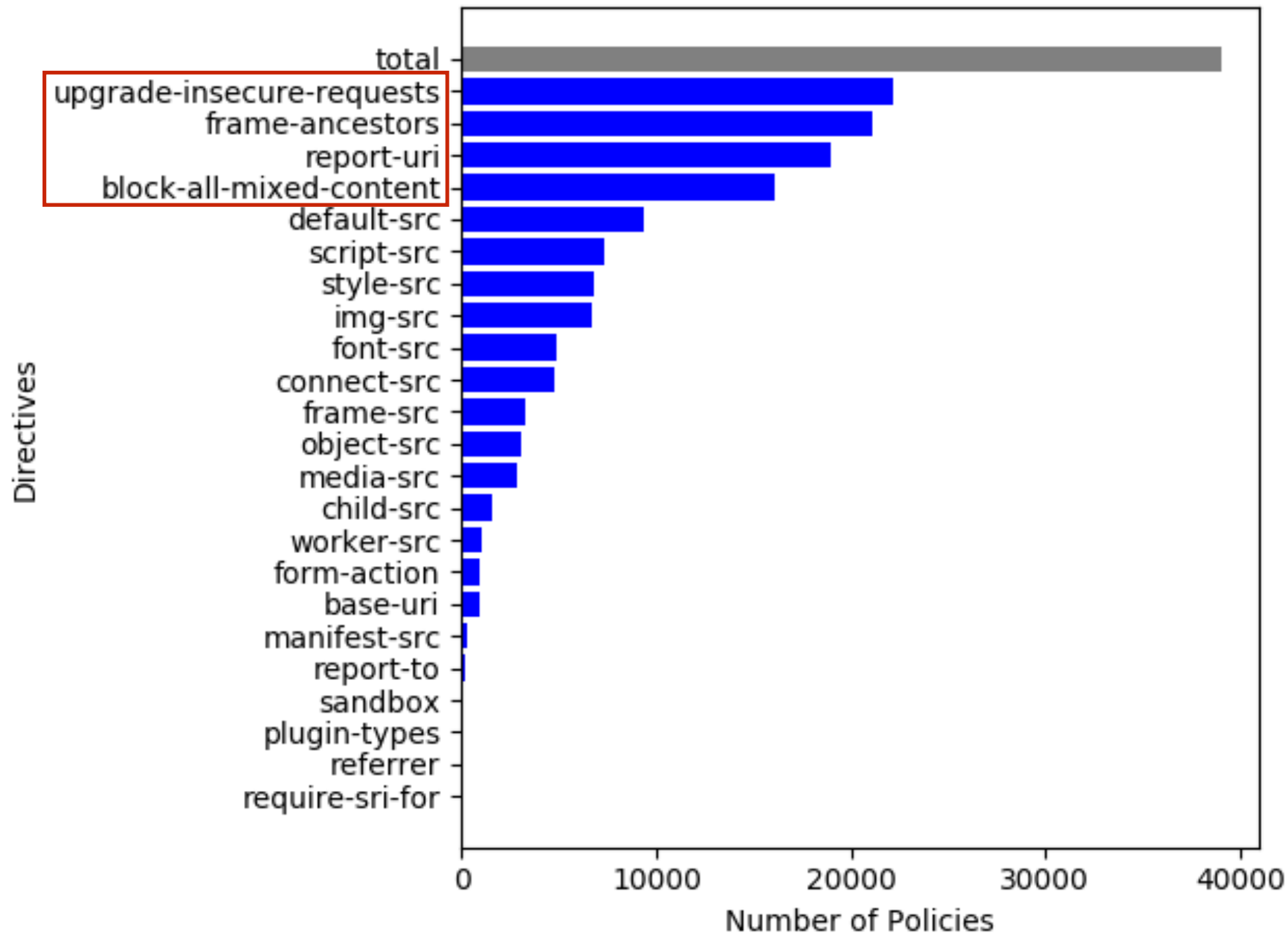
Other adoption challenges

- Goal: set most restricting CSP that is permissive enough to not break existing app
- How can you figure this out for a large app?
  - CSP has a report-only header and report-uri directive
  - Report violations to server; don't enforce
- In practice: devs hardly ever get the policy right

How is CSP really used in practice?

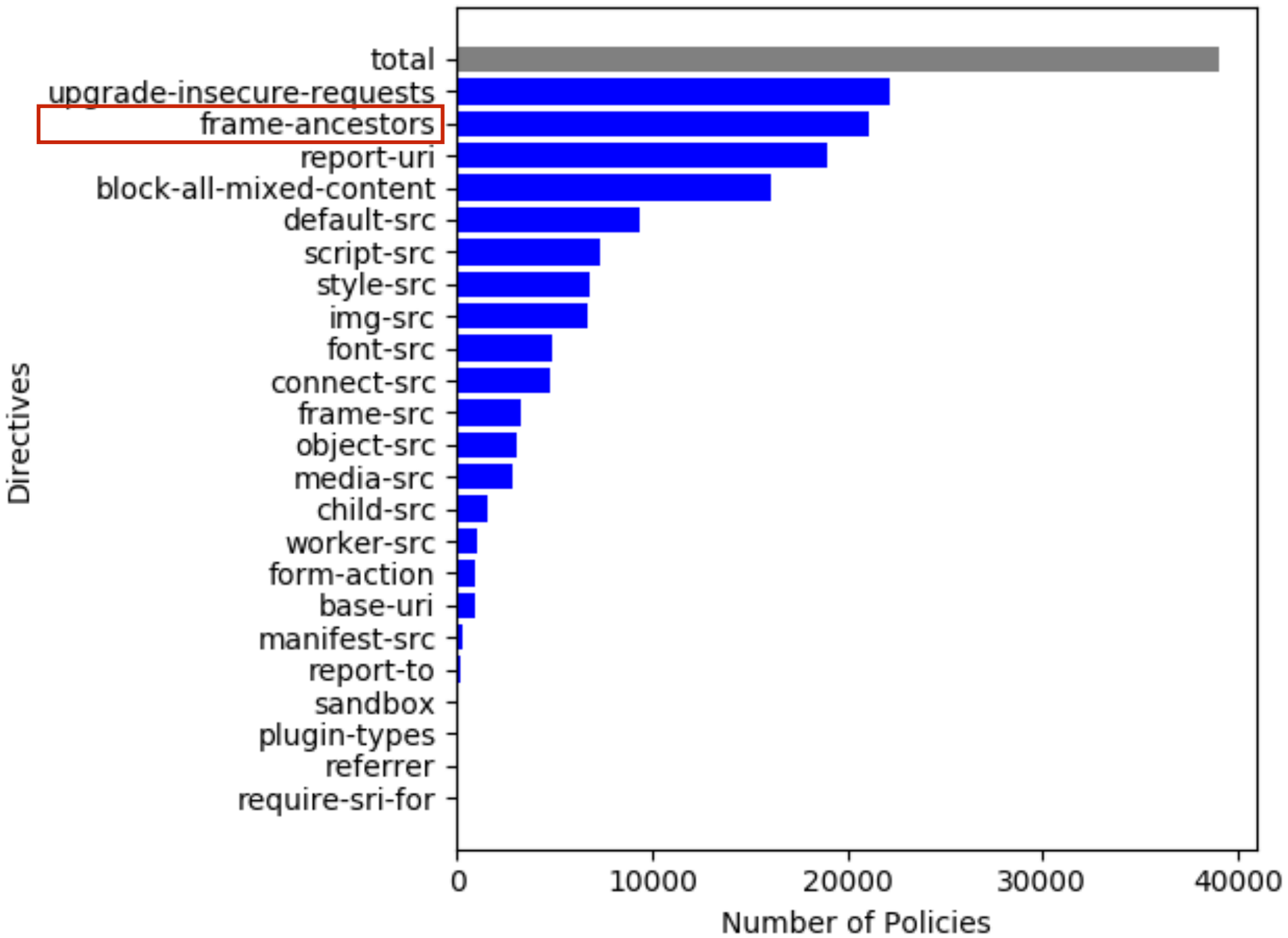


How is CSP really used in practice?



Warning: This graph is a few years old

How is CSP really used in practice?



What's frame-ancestors?

The HTTP `Content-Security-Policy` (CSP) **`frame-ancestors`** directive specifies valid parents that may embed a page using `<frame>`, `<iframe>`, `<object>`, `<embed>`, or `<applet>`.

Setting this directive to `'none'` is similar to `X-Frame-Options: deny` (which is also supported in older browers).

What problem is this addressing?

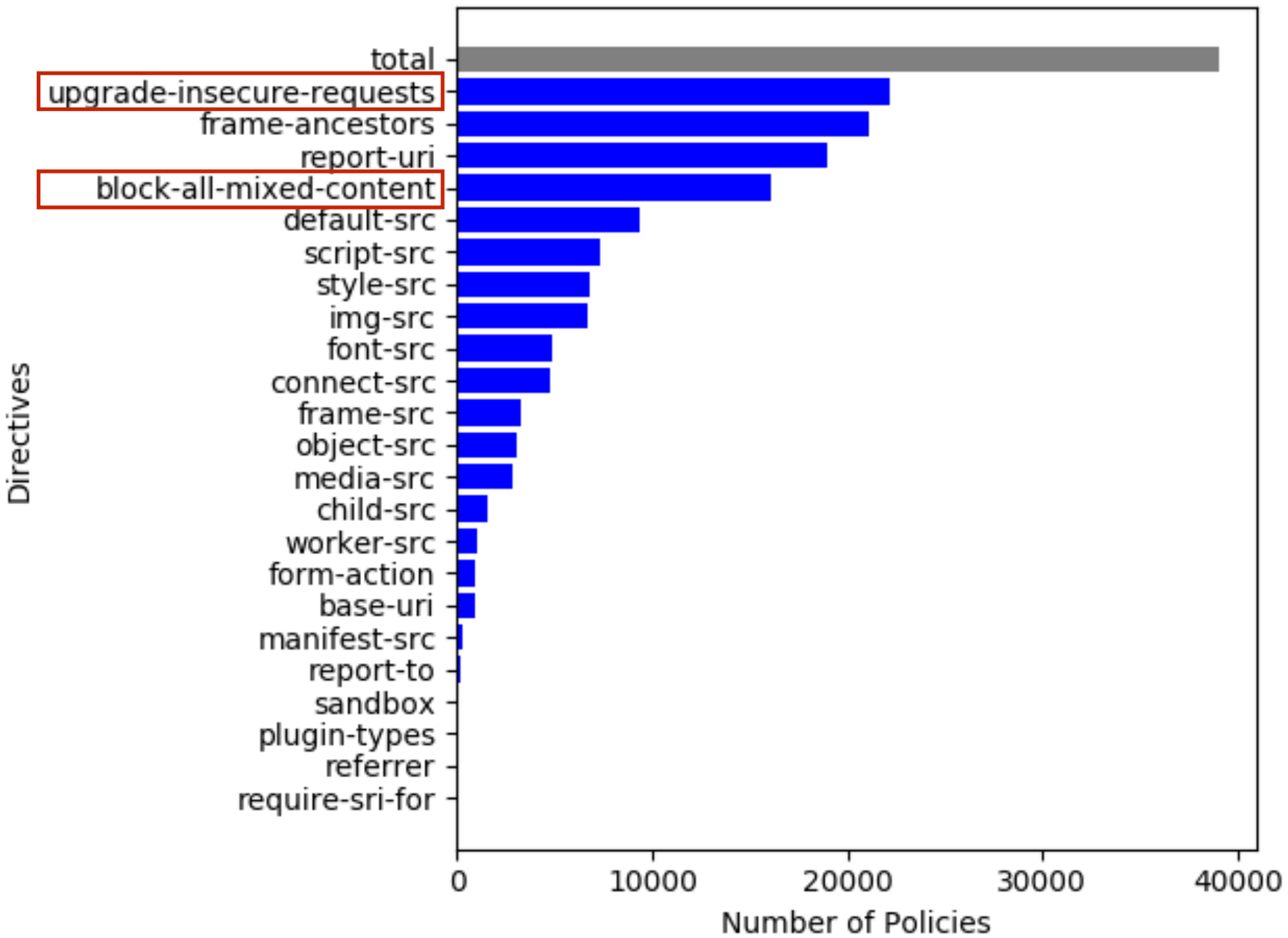
Clickjacking!

- How does frame-ancestor help?
  - Don’t allow non twitter origins to frame delete page!

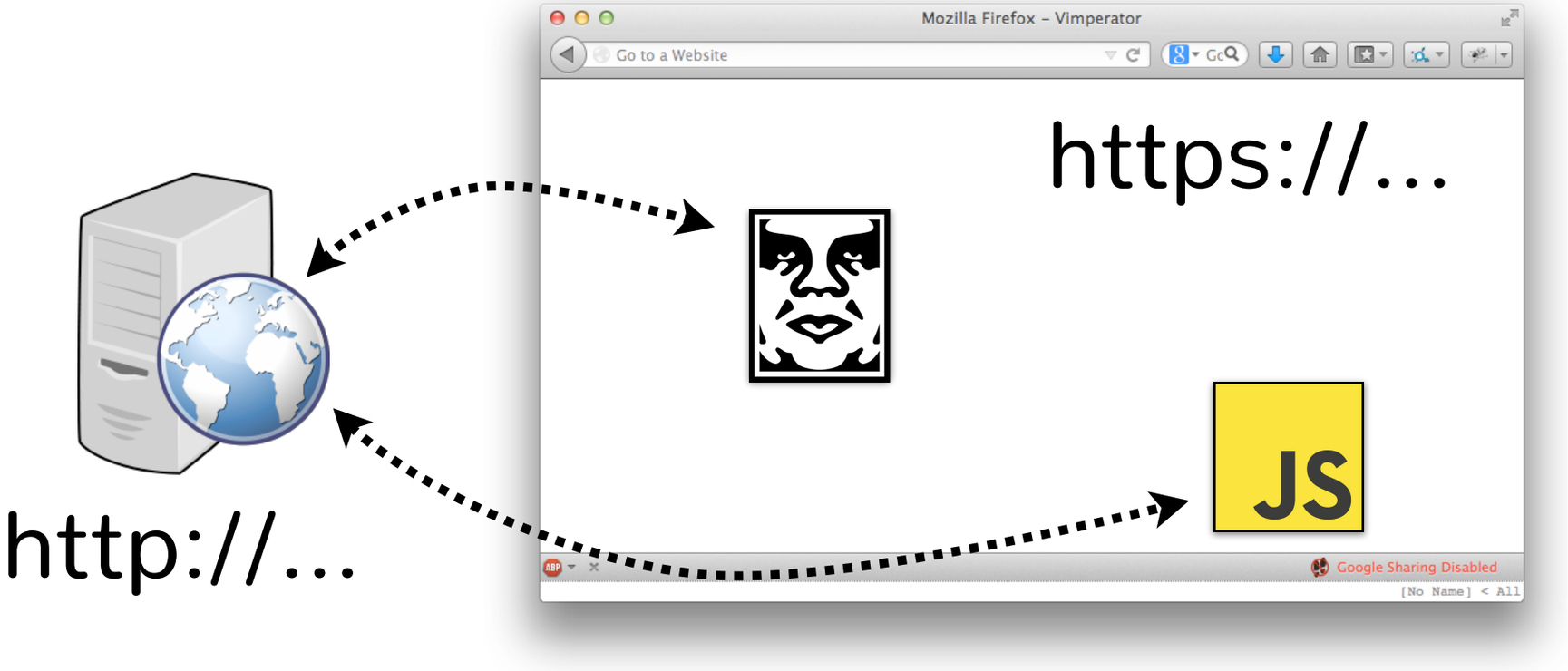
http://best.game.ever

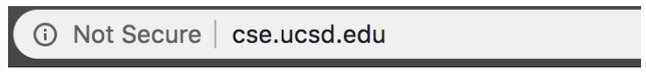
twitter.com

What about the other two?



What is MIXed content?

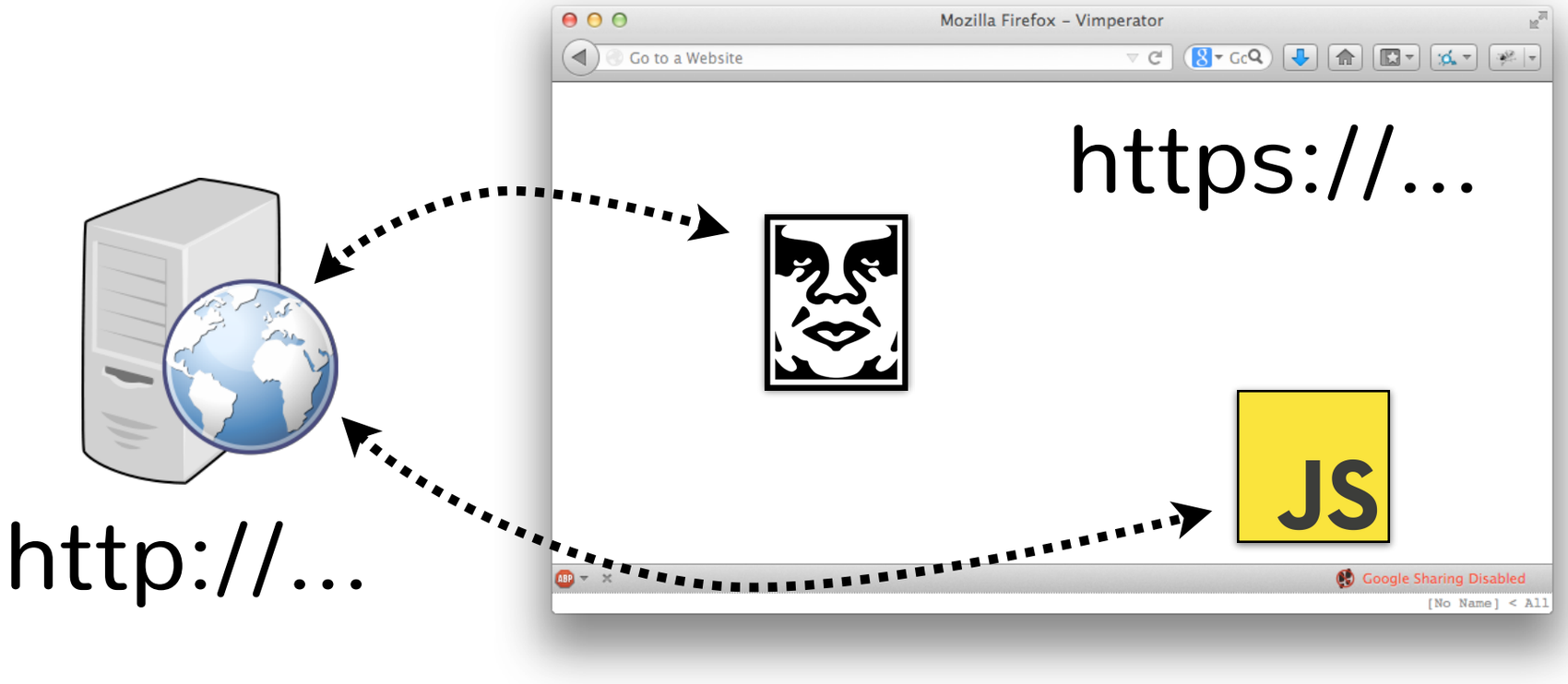


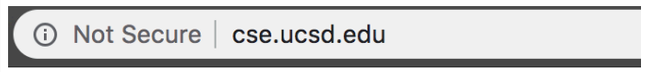


Computer Science and Engineering

- Why is this bad?

What is MIXed content?





Computer Science and Engineering

- Why is this bad?
  - Network attacker can inject their own scripts, images, etc.!

How does CSP help?

- upgrade-insecure-requests
  - Rewrite HTTP URL to HTTPS before making request
- block-all-mixed-content
  - Don’t load any content over HTTP

Outline: modern mechanisms

- iframe sandbox (quick refresher)
- Content security policy (CSP)
- HTTP strict transport security (HSTS)
- Subresource integrity (SRI)
- Cross-origin resource sharing (CORS)

Motivation for HSTS

- Attacker can force you to go to HTTP vs. HTTPS
  - SSL Stripping attack (Moxie)
    - They can rewrite all HTTPS URLs to HTTP
    - If server serves content over HTTP: doom!
- **HSTS header: never visit site over HTTP again**
  - Strict-Transport-Security: max-age=31536000

Outline: modern mechanisms

- iframe sandbox (quick refresher)
- Content security policy (CSP)
- HTTP strict transport security (HSTS)
- → Subresource integrity (SRI)
- Cross-origin resource sharing (CORS)

Motivation for SRI

- CSP+HSTS can be used to limit damages, but can't really defend against malicious code
- How do you know that the library you're loading is the correct one?

Won't using HTTPS address this problem?

- CSP+
can't r
- How
is the
Won't

jQuery
drive-

MaxCDN

Date: 2013-07-02

MaxCDN, a content-delivery network service had their servers compromised. MaxCDN is running bootstrapcdn.com, a CDN download for popular Bootstrap front end framework.

The vendor of MaxCDN had laid off a support engineer having access to the servers where BootstrapCDN runs. The credentials of the support engineer were not properly revoked. The attackers had gained access to these credentials. The attackers rebooted the server into single-user mode, changed the root password, and SSH'd into the server. Bootstrap JavaScript files were modified to serve an exploit toolkit.

Bootstrap is widely deployed and CDN option is one of the recommended ways to include Bootstrap on your website. BootstrapCDN gets a lot of downloads. Thus, the attack payload was served to tens of thousands of visitors in short period of time.

Related evaluation points:

- Passphrase on server login keys
- Audited server login keys
- HTTPS / TLS only

Links:

- BootstrapCDN Security Post-Mortem

Mikko Ohtamaa

Subresource integrity

- Idea: page author specifies hash of (sub)resource they are loading; browser checks integrity
  - E.g., integrity for scripts

```
<link rel="stylesheet" href="https://site53.cdn.net/style.css"
      integrity="sha256-SDfwewFAE...wefjijfE">
```

  - E.g., integrity for link elements

```
<script src="https://code.jquery.com/jquery-1.10.2.min.js"
   integrity="sha256-C6CB9UYIS9UJeqinPHWTHVqh/E1uhG5Tw+Y5qFQmYg=">
```

What happens when check fails?

- Case 1 (default):
  - Browser reports violation and does not render/execute resource
- Case 2: CSP directive with integrity-policy directive set to report
  - Browser reports violation, but may render/execute resource

Multiple hash algorithms

- Authors may specify multiple hashes
  - E.g., `<script src="hello_world.js" integrity="sha256-... sha512-..."></script>`
- Browser uses strongest algorithm
- Why support multiple algorithms?
  - Don't break page on old browser

Outline: modern mechanisms

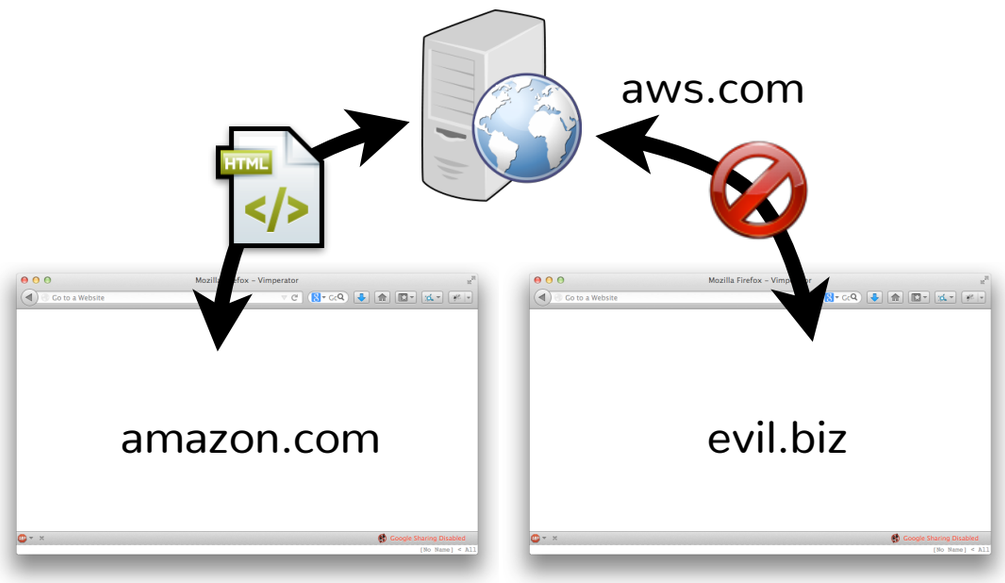
- iframe sandbox (quick refresher)
- Content security policy (CSP)
- HTTP strict transport security (HSTS)
- Subresource integrity (SRI)
- → Cross-origin resource sharing (CORS)

Recall: SOP is also inflexible

- Problem: Can't fetch cross-origin data
  - Leads to building insecure sites/services: JSONP
- Solution: Cross-origin resource sharing (CORS)
  - Data provider explicitly whitelists origins that can inspect responses
  - Browser allows page to inspect response if its origin is listed in the header

E.g., CORS usage: amazon

- Amazon has multiple domains
  - E.g., amazon.com and aws.com
- Problem: amazon.com can't read cross-origin aws.com data
- With CORS aws.com can allow amazon.com to read HTTP responses



How CORS works

- Browser sends Origin header with XHR request
  - E.g., Origin: https://amazon.com
- Server can inspect Origin header and respond with Access-Control-Allow-Origin header
  - E.g., Access-Control-Allow-Origin: https://amazon.com
  - E.g., Access-Control-Allow-Origin: *
- CORS XHR may send cookies + custom headers
  - Need “preflight” request to authorize this

New: Permissions Policy

- Web platform has access to many things
  - accelerometer, ambient-light-sensor, battery, camera, display-capture, geolocation, ...
- Permissions policy tries to limit access to certain resources
  - Header:
    `Permissions-Policy: fullscreen=(), geolocation=()`
  - Iframe attribute:
    `<iframe src="https://other.com/map" allow="geolocation"></iframe>`

- ✓ How do we protect page from ads/services?
- ✓ How to share data with cross-origin page?
- ✓ How to protect one user from another's content?
- ✓ How do we protect the page from a library?
- ✓ How do we protect the page from the CDN?
- ✓ How do we protect the page from network provider?