HTTP Security Headers

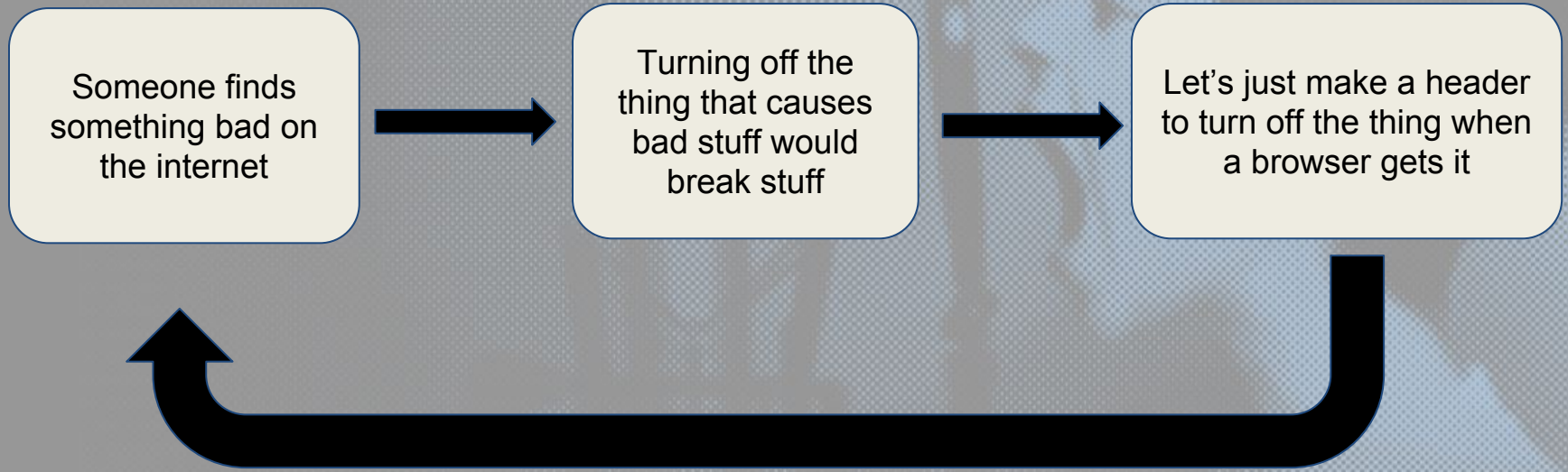
A technology history through scar tissue

Benjamin Hering
inbox@benjamin-hering.com

All Day DevOps

October 17, 2018 - @SecTinkerer

Security Header Flowchart

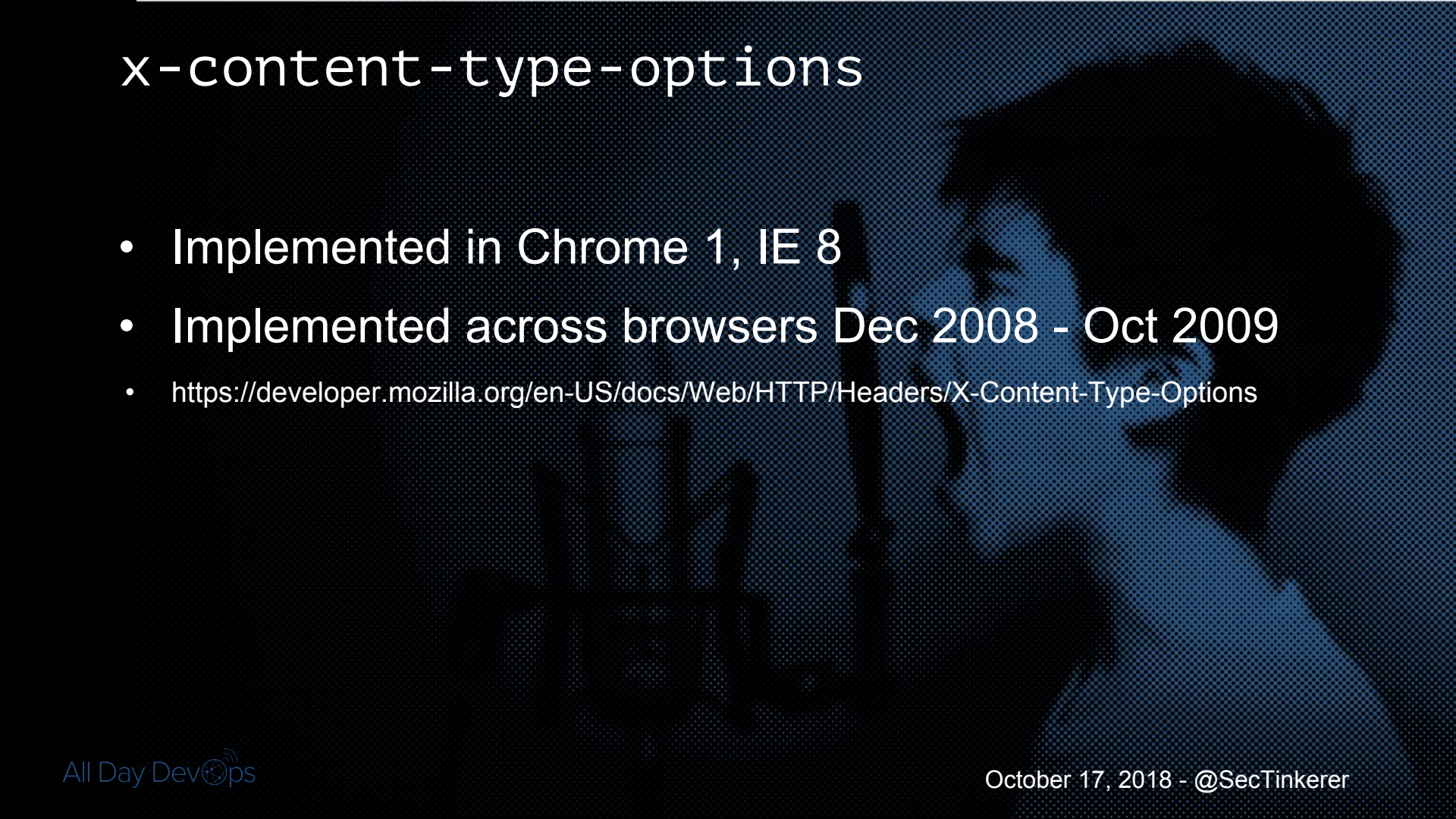


Headers we'll cover

- `x-content-type-options`
- `x-xss-protection`
- `x-frame-options`
- `strict-transport-security`
- `referrer-policy`
- `feature-policy` (kinda, it's not quite here yet)
- `content-security-policy`
- Managing CSPs at LendingClub

x-content-type-options

- Implemented in Chrome 1, IE 8
- Implemented across browsers Dec 2008 - Oct 2009
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options



What's the bad thing?

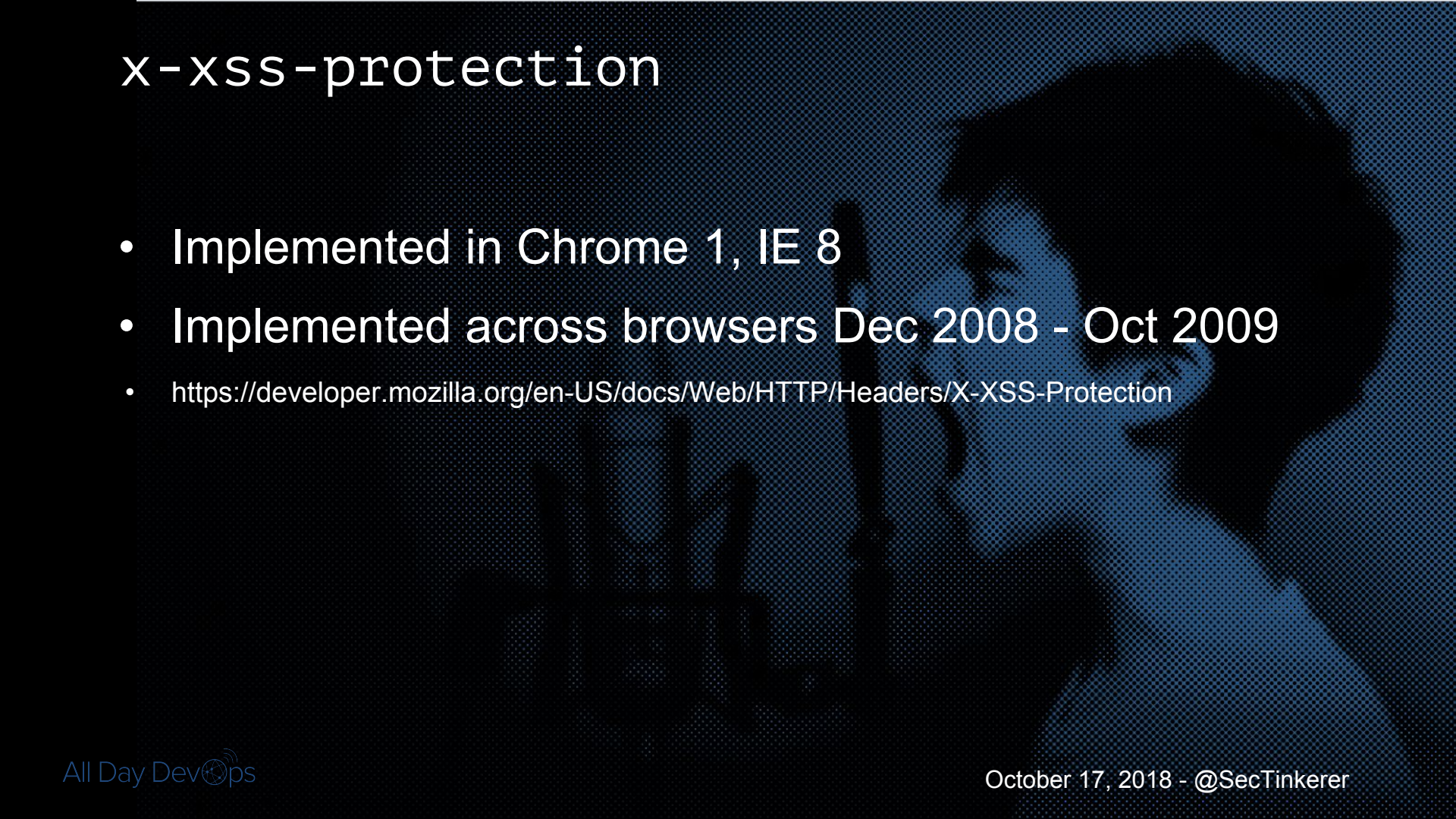
- Upload executable files with a non-executable MIME type
- Attacker tricks upload of malicious JavaScript as foo.txt (`text/plain` MIME)
- Browser 'sniffs' the file as actually JavaScript. Changes to `application/javascript` MIME and executes

Set to `x-content-type-options: nosniff`

Tells browsers to not try and guess MIME types of files

Just set your MIME types correctly

x-xss-protection



- Implemented in Chrome 1, IE 8
- Implemented across browsers Dec 2008 - Oct 2009
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

What's the bad thing?

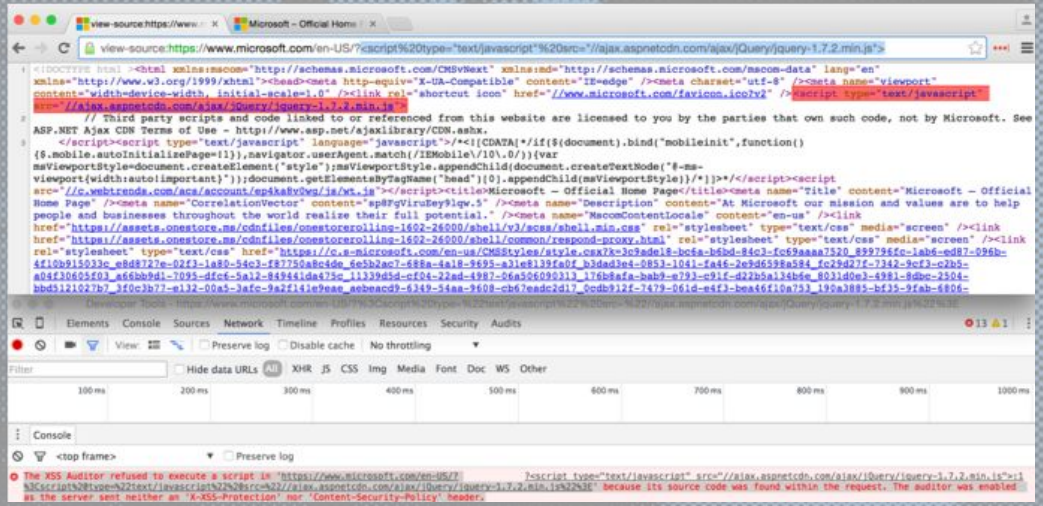
- Reflected Cross Site Scripting (XSS)
- Rolled out in IE 8, off by default could be enabled by headers
- Stops the lowest hanging XSS fruit

```
GET /index.php&name=<script>badstuff</script>

<html>
…
<body>
<p>Hello<script>badstuff</script>!
</p>
…
```

What to set?

- `x-xss-protection: 1`
  - Turns on the filter (already on by default in most browsers). Kinda silly.
- `x-xss-protection: 1; mode=block`
  - Instead of stripping out what it thinks, blocks the entire request
  - MS10-002 abused the filter to perform XSS attacks through the XSS filter
  - XSS filter can be used to knock out legit scripts
  - **CSP header is better**

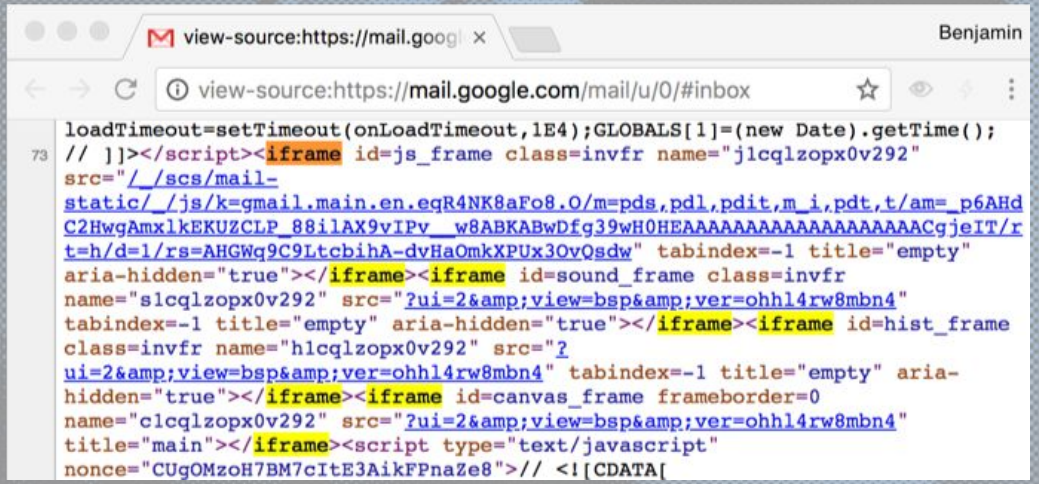


x-frame-options

- Implemented in Chrome 4, Firefox 3.6.9, IE 8
- Implemented across browsers Oct 2009 – Sept 2010
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

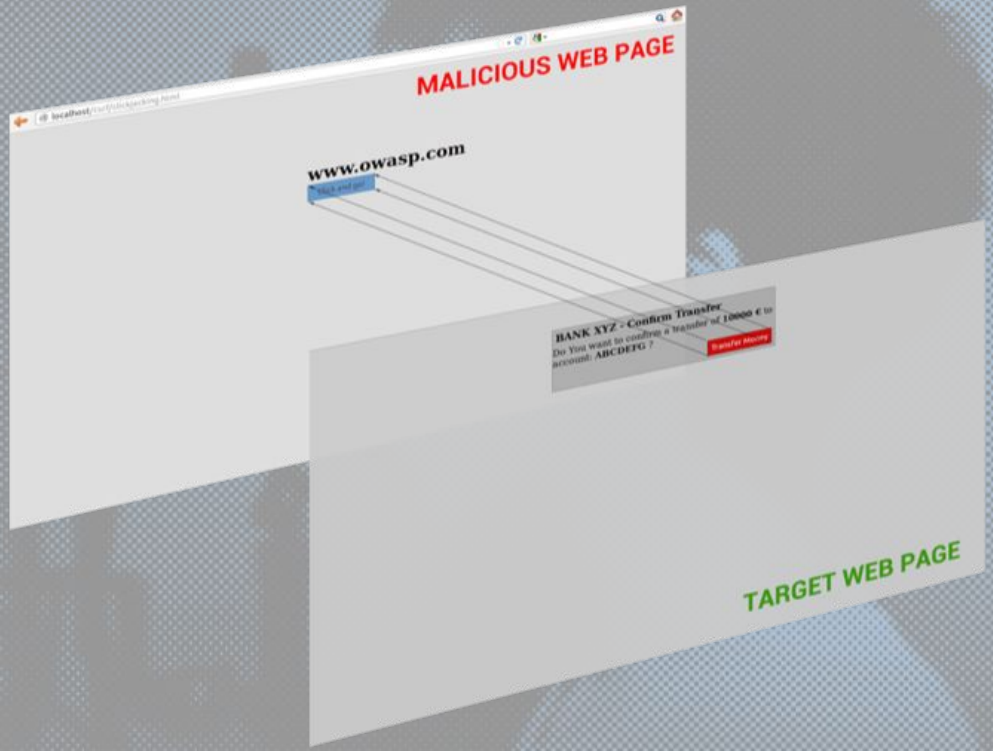
<frame> (deprecated) <iframe> <object>

- Embeds pages in other pages
- <frame> is obsolete in HTML5, but
- <iframe> is still in gmail and will probably never die



What's the bad thing?

- Clickjacking
- Drawing a cosmetic top above actual buttons
- Add keystroke logging JS



What to set?

- `x-frame-options: DENY`
  - No embedding at all
- `x-frame-options: SAMEORIGIN`
  - Allow embedding from the same origin as embedded page
- `x-frame-options: ALLOW-FROM https://example.com`
  - Allow embedding only on specifically outlined locations

(`content-security-policy: frame-ancestors 'none';` also works, but only in some modern browsers. No IE, No Microsoft Edge on mobile)

`https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/frame-ancestors`

strict-transport-security

- Implemented in Chrome 4, Firefox 4, IE 11
- Implemented across browsers January 2010 – October 2013
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

What's the bad thing?

Where's example.com?

example.com is at 42.42.42.42

Great. Give me http://example.com 42.42.42.42 port 80

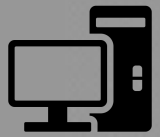
Let's do HTTPS instead. 301 Redirect https://example.com

No problem. Give me https://example.com 42.42.42.42 port 443

Here's my TLS cert

I trust that cert. I'll load the page

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta content=..



What's the bad thing?

Where's example.com?

example.com happens to be me! 192.168.1.1

Great. Give me http://example.com 192.168.1.1 port 80

Give me https://example.com 42.42.42.42 port 443

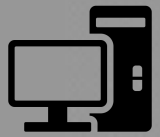
Here's my TLS cert

Sure whatever, give me the page

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><script src="evil.com/bad.js" …

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta content=..

Sweet. Lemme just go run evil.com/bad.js



HTTP

HTTPS

What's the bad thing?

Where's example.com?

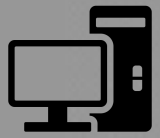
example.com happens to be me! 192.168.1.1

Great. Give me http://example.com 192.168.1.1 port 80

Wait, example.com has HSTS enabled! Give me https://example.com 192.168.1.1 port 443

Uh, okay, here's my cert

This cert is junk! Don't load!



HTTPS

What to set?

- `strict-transport-security: max-age=<expire-time>`
  - All further requests to that site will be over HTTPS until age expires
- `strict-transport-security: max-age=<expire-time>; includeSubDomains`
  - All further requests to that site & subdomains will be over HTTPS until age expires
- `strict-transport-security: max-age=<expire-time>; includeSubDomains; preload`
  - All further requests to that site & subdomains will be over HTTPS until age expires and it will get preloaded into browsers
  - Preload requires:
    - `max-age=31536000` or higher (one year)
    - `includeSubDomains`
    - valid HTTPS cert, serving HTTPS on all subdomains
    - redirect port 80 to port 443 (if listening on port 80)
  - If any subdomain can't handle HTTPS **it will break**. By design, no quick rollbacks.
  - https://hstspreload.org/

referrer-policy

- Implemented in Chrome 61, Firefox 52, no Safari or Microsoft Edge support
- Implementation started in browsers March 2017
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy

What's the bad thing?

- When someone clicks a link off your site, you can leak specific information about a user in the "referer" (sic) header
- Maybe your site is sensitive
- Or specific URLs can be used to identify users

```
referer:_https://www.lendingclub.com/account/
loanPerf.action?loan_id=88783584&order_id=124
783505&note_id=131056825

referer: https://site-govt-doesnt-like.com

referer: https://twitter.com/SecTinkerer
```

What to set?

- Depends on your site. Common good choices are:
- `referrer-policy: strict-origin`
  - Only sends the origin and not URI info. Won't send over HTTP (`referer: https://www.lendingclub.com`)
- `referrer-policy: strict-origin-when-cross-origin`
  - Only sends origin as a referer when links off your site. Internally, the header keeps the full uri. Won't send the header over HTTP
- `referrer-policy: no-referrer`
  - Just don't send the referer header ever

feature-policy

- Just Chrome; no IE, Edge, Firefox or Safari support
- New this year
- Allows or denies browser features
- “Like CSP for features!”

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Feature-Policy
- https://developers.google.com/web/updates/2018/06/feature-policy

Basic support	60	No	No	No	47	No	60	60	No	No	47	No	No

content-security-policy

- Implemented in Chrome 25, Firefox 23, IE 10
- Implemented across browsers Sept 2012– April 2013
- Supersedes the previous x-xss-protection, x-frame-options headers
- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

What’s the bad thing?

- Somewhere, somehow, something on your site that you don’t want running is there
- CSP let’s you lock down content that page can load to a whitelist
- And it will report back to you anything that’s in violation

Restrict any content!

- URLs that are loaded with script interfaces
- Fonts
- Frames
- Form submission endpoints
- What plugins can run
- Images
- Manifest files
- What <audio> <video> and <track> elements can load
- What <object>, <embed> and <applet> elements can load
- JavaScript
- CSS
- And defaults for everything else

Build from the ground up

- `content-security-policy: default-src 'self'; report-uri /csp/report`
  - Only load content from the same origin (no subdomains!) anything else send a report
- `content-security-policy: default-src 'self'; connect-src ‘self’ *.optimizely.com; report-uri /csp/report`
  - Adds the ability to run connect to any optimizely.com subdomain
- `content-security-policy: default-src 'self'; connect-src ‘self’ *.optimizely.com; script-src 'unsafe-inline' 'self' js-agent.newrelic.com; report-uri /csp/report`
  - Adds the ability to run inline JavaScript and scripts from js-agent.newrelic.com

What's this in real life? (news.google.com)

- Specifying a nonce (a unique per request, base64 value string) makes modern browsers ignore "unsafe-inline"

  `<script nonce="aWQGtiBJUDdHaq1L6xfcPaQBMeA"> … </script>`

- Yes, you can have multiple content-security-policy headers. Additions can only restrict. Most strict policy wins.

What's this in real life? (facebook.com)

```
content-security-policy: default-src * data: blob:;script-src *.facebook.com *.fbcdn.net *.facebo
ok.net *.google-analytics.com *.virtualearth.net *.google.com 127.0.0.1:* *.spotilocal.com:*
'unsafe-inline' 'unsafe-eval' *.atlassolutions.com blob: data: 'self';style-src data: blob:
'unsafe-inline' *;connect-src *.facebook.com facebook.com *.fbcdn.net *.facebook.net *.spotil
ocal.com:* wss://*.facebook.com:* https://fb.scanandcleanlocal.com:* *.atlassolutions.com att
achment.fbsbx.com ws://localhost:* blob: *.cdninstagram.com 'self' chrome-extension://boadgeo
jelhgndaghljhdicfkmllpafd chrome-extension://dliochdbjfkdbacpmhlcpmleaejidimm;
```

What's this in real life? (twitter.com)

content-security-policy: script-src https://connect.facebook.net https://cm.g.doubleclick.net https://ssl.google-analytics.com https://graph.facebook.com 'nonce-eYyW3aaTY/WRlhD2P/SGBQ==' https://twitter.com 'unsafe-eval' https://*.twimg.com https://api.twitter.com https://analytics.twitter.com https://publish.twitter.com https://ton.twitter.com https://syndication.twitter.com https://www.google.com https://t.tellapart.com https://platform.twitter.com https://www.google-analytics.com blob: 'self'; frame-ancestors 'self'; font-src https://twitter.com https://*.twimg.com data: https://ton.twitter.com https://fonts.gstatic.com https://maxcdn.bootstrapcdn.com https://netdna.bootstrapcdn.com 'self'; media-src https://rmpdhdsnappytv-vh.akamaihd.net https://prod-video-eu-central-1.pscp.tv https://prod-video-ap-south-1.pscp.tv https://v.cdn.vine.co https://dwo3ckksxlb0v.cloudfront.net https://twitter.com https://prod-video-us-east-2.pscp.tv https://prod-video-cn-north-1.pscp.tv https://amp.twimg.com https://smmdhdsnappytv-vh.akamaihd.net https://*.twimg.com https://prod-video-eu-west-1.pscp.tv https://*.video.pscp.tv https://rmmdhdsnappytv-vh.akamaihd.net https://clips-media-assets.twitch.tv https://prod-video-ap-northeast-2.pscp.tv https://prod-video-us-west-2.pscp.tv https://prod-video-us-west-1.pscp.tv https://prod-video-ap-northeast-1.pscp.tv https://smdhdsnappytv-vh.akamaihd.net https://ton.twitter.com https://prod-video-eu-west-3.pscp.tv https://rmdhdsnappytv-vh.akamaihd.net https://mmdhdsnappytv-vh.akamaihd.net https://prod-video-ca-central-1.pscp.tv https://smpdhdsnappytv-vh.akamaihd.net https://prod-video-sa-east-1.pscp.tv https://mdhdsnappytv-vh.akamaihd.net https://prod-video-ap-southeast-2.pscp.tv https://mtc.cdn.vine.co https://prod-video-cn-northwest-1.pscp.tv https://prod-video-eu-west-2.pscp.tv https://dev-video-us-west-2.pscp.tv https://prod-video-us-east-1.pscp.tv blob: 'self' https://prod-video-ap-northeast-3.pscp.tv https://prod-video-ap-southeast-1.pscp.tv https://mpdhdsnappytv-vh.akamaihd.net https://dev-video-eu-west-1.pscp.tv; connect-src https://rmpdhdsnappytv-vh.akamaihd.net https://prod-video-eu-central-1.pscp.tv https://graph.facebook.com https://prod-video-ap-south-1.pscp.tv https://*.giphy.com https://dwo3ckksxlb0v.cloudfront.net https://prod-video-us-east-2.pscp.tv https://prod-video-cn-north-1.pscp.tv https://vmaprel.snappytv.com https://smmdhdsnappytv-vh.akamaihd.net https://*.twimg.com https://embed.pscp.tv https://api.twitter.com https://prod-video-eu-west-1.pscp.tv https://*.video.pscp.tv https://rmmdhdsnappytv-vh.akamaihd.net https://clips-media-assets.twitch.tv https://prod-video-ap-northeast-2.pscp.tv https://prod-video-us-west-2.pscp.tv https://pay.twitter.com https://prod-video-us-west-1.pscp.tv https://analytics.twitter.com https://vmap.snappytv.com https://*.twprobe.net https://prod-video-ap-northeast-1.pscp.tv https://smdhdsnappytv-vh.akamaihd.net https://prod-video-eu-west-3.pscp.tv https://syndication.twitter.com https://sentry.io https://rmdhdsnappytv-vh.akamaihd.net https://media.riffsy.com https://mmdhdsnappytv-vh.akamaihd.net https://prod-video-ca-central-1.pscp.tv https://embed.periscope.tv https://smpdhdsnappytv-vh.akamaihd.net https://prod-video-sa-east-1.pscp.tv https://vmapstage.snappytv.com https://upload.twitter.com https://proxsee.pscp.tv https://mdhdsnappytv-vh.akamaihd.net https://prod-video-ap-southeast-2.pscp.tv https://prod-video-cn-northwest-1.pscp.tv https://prod-video-eu-west-2.pscp.tv https://dev-video-us-west-2.pscp.tv https://prod-video-us-east-1.pscp.tv 'self' https://prod-video-ap-northeast-3.pscp.tv https://vmap.grabyo.com https://prod-video-ap-southeast-1.pscp.tv https://mpdhdsnappytv-vh.akamaihd.net https://dev-video-eu-west-1.pscp.tv; style-src https://fonts.googleapis.com https://twitter.com https://*.twimg.com https://translate.googleapis.com https://ton.twitter.com 'unsafe-inline' https://platform.twitter.com https://maxcdn.bootstrapcdn.com https://netdna.bootstrapcdn.com 'self'; object-src https://twitter.com https://pbs.twimg.com; default-src 'self' blob:; frame-src https://staticxx.facebook.com https://twitter.com https://*.twimg.com https://5415703.fls.doubleclick.net https://player.vimeo.com https://pay.twitter.com https://www.facebook.com https://ton.twitter.com https://syndication.twitter.com https://vine.co twitter: https://www.youtube.com https://platform.twitter.com https://upload.twitter.com https://s-static.ak.facebook.com https://4337974.fls.doubleclick.net https://8122179.fls.doubleclick.net 'self' https://donate.twitter.com; img-src https://graph.facebook.com https://*.giphy.com https://*.pscp.tv https://twitter.com https://*.twimg.com https://ad.doubleclick.net data: https://clips-media-assets.twitch.tv https://lumiere-a.akamaihd.net https://fbcdn-profile-a.akamaihd.net https://www.facebook.com https://ton.twitter.com https://*.fbcdn.net https://syndication.twitter.com https://media.riffsy.com https://www.google.com https://stats.g.doubleclick.net https://platform.twitter.com https://api.mapbox.com https://www.google-analytics.com blob: https://*.periscope.tv 'self'; report-uri https://twitter.com/i/csp_report?a=NVQWGYLXFVZXO2LGOQ%3D%3D%3D%3D%3D%3D%3D&ro=false;

Managing `content-security-policy` at LendingClub

- Empowering developers through a layered hierarchy of control
- Setting global fallbacks for legacy applications
- Leveraging `content-security-policy-report-only`

Developer led CSPs

Real time config

- Can overwrite both app specific and global values
- Requires change control approval to differ from static

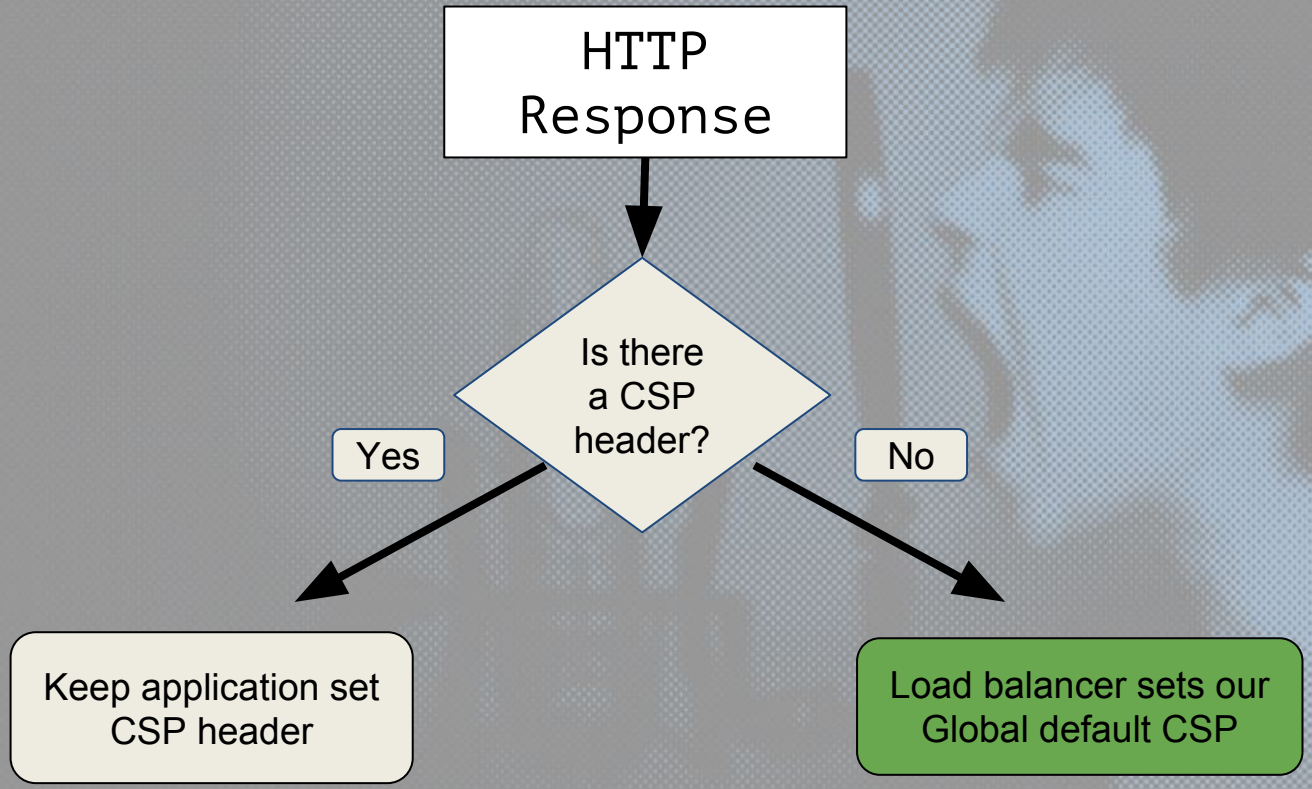
Application static config

- Only application specific additions (loosening)
- Standard code review and security team informed

Global defaults

- reporting uri
- analytics, A/B test vendor, etc
- Requires committee approval to change

What about legacy stuff?



`content-security-policy-report-only`

- Basically “audit mode” for `content-security-policy`
- Displays violations in the browser console, but won’t block functionality
- Syntax is identical; only appending `-report-only`
- Reports violations via report-uri
  - Can be helpful to have a different reporting URI for each

Example Report

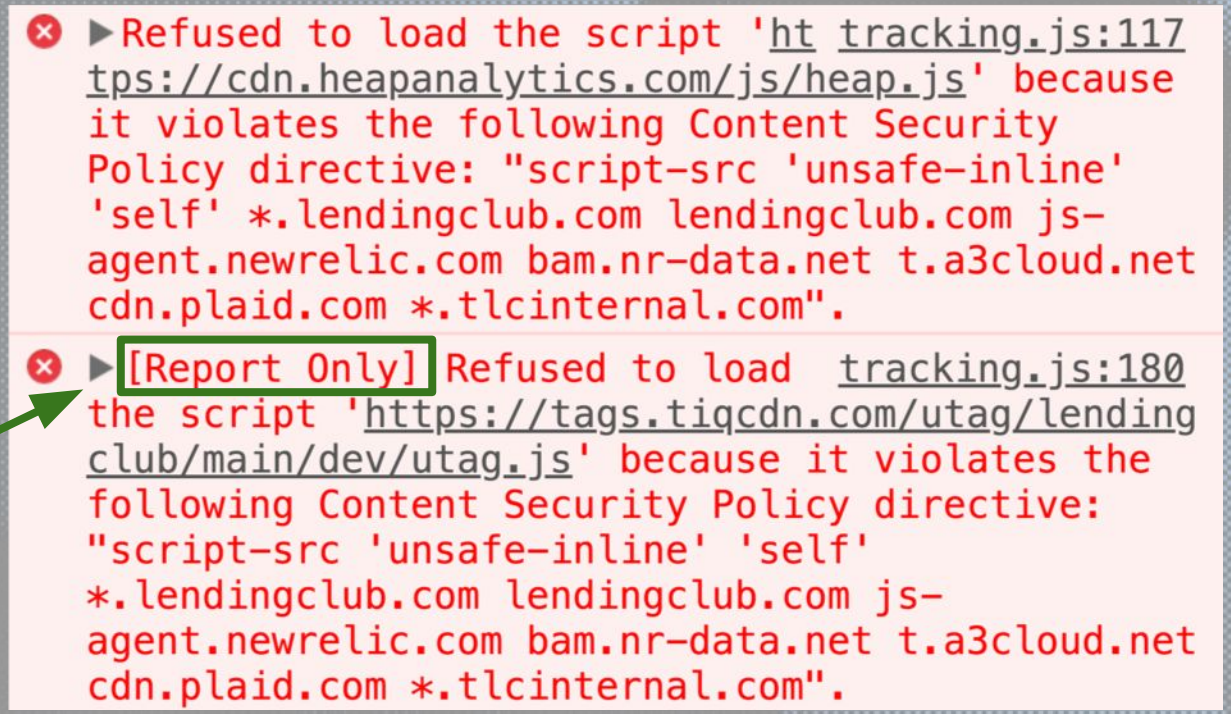
JSON formatted POST to your report URI

```
{
  "csp-report": {
    "document-uri": "http://example.com/signup.html",
    "referrer": "",
    "blocked-uri": "http://example.com/css/style.css",
    "violated-directive": "style-src cdn.example.com",
    "original-policy": "default-src 'none'; style-src
cdn.example.com; report-uri /_/csp-reports",
    "disposition": "report/enforce"
  }
}
```

Example Console Errors

A blocked CSP violation

A non-blocking CSP-RO violation



Takeaways from LendingClub

- Make the lowest friction way to change CSPs your normal path to test and push code to production
  - Still have an emergency break for real-time changes
  - Still have a consensus global baseline for legacy apps
- Use the `-report-only` header to sanity check changes
- DevOps & Developer led CSP settings are faster, safer and stronger than top-down security mandates

Thank You All Day DevOps Sponsors

Platinum Sponsors

Sonatype

cloudbees

Gold Sponsors

puppet

GitLab

XebiaLabs
Enterprise DevOps

GENERAL DYNAMICS
Information Technology

cisco DEVNET
developer.cisco.com

Carnegie Mellon University
Software Engineering Institute

CHEF

SCALED AGILE

Media Sponsors

TechBeacon

DevOps.com

DZone

Solutions Review

Thank You All Day DevOps Supporters

Meet me in the Slack channel for Q&A

bit.ly/addo-slack

#2018addo-devsecops channel

Icons from the Noun Project
desktop by Markus from the Noun Project
modem by Markus from the Noun Project
Server by Mister Pixel from the Noun Project
Phishing attack by ProSymbols from the Noun Project

All Day DevOps

Benjamin Hering
inbox@benjamin-hering.com

October 17, 2018 - @SecTinkerer