HTTP Security Headers You Need To Have On Your Web Apps

Slides up at scottsauber.com

scottsauber

Audience

- Anyone with a web app

scottsauber

Agenda

- What are HTTP Security Headers?
- Why do they matter?
- HSTS, XFO, XSS, CSP, CTO, RH, FP, PP
  - What are they
  - What do they do
  - Demo
  - Impact on existing apps

Goals

- Expose you to security headers that are out there
- Why they are needed
- Write down ones you need to look into when you’re back at work

Who am I?

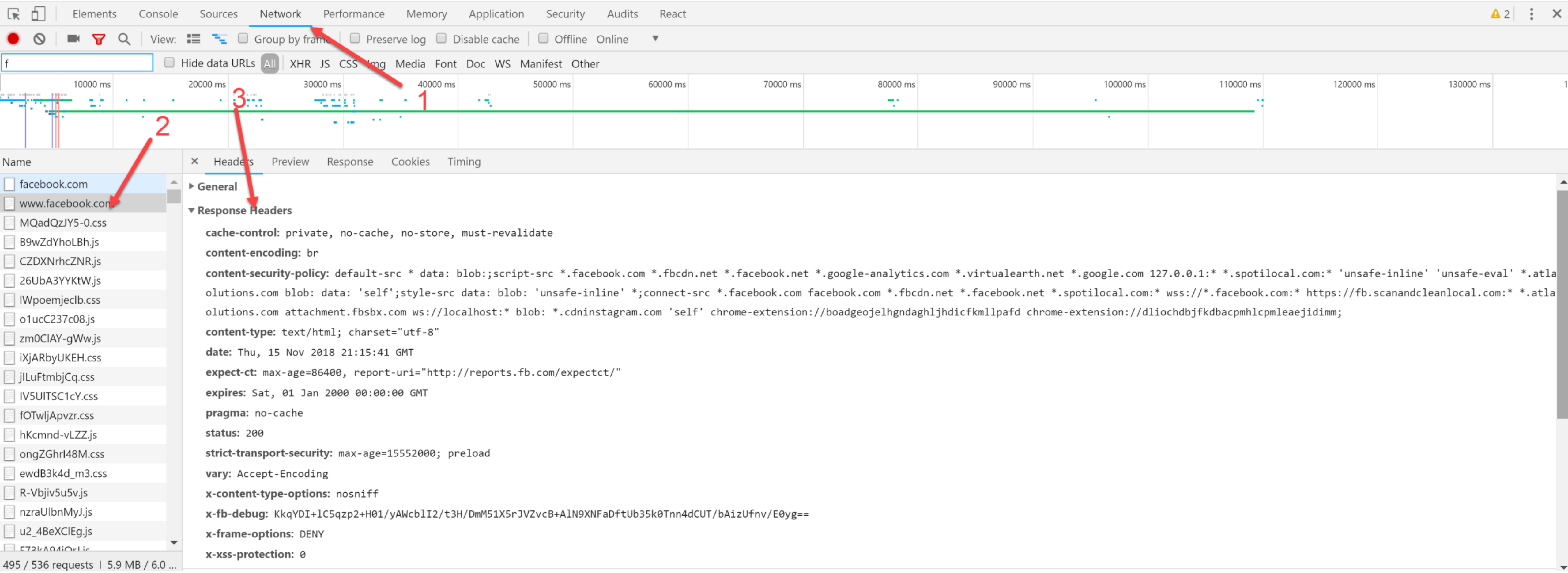
- Director of Engineering at Lean TECHniques
- Co-organizer of Iowa .NET User Group
- Microsoft MVP
- Friend of Redgate
- Blog at scottsauber.com
- Not a security expert… but…

What are HTTP Headers?

- Allows both the client and server to pass additional data along to the request or response to exchange information and inform the other party.
- Request header examples:
  - Cookies
  - Accept-language: en-us
- Response header examples:
  - Date
  - Content-type: text/html or application/json
  - ***Security-related headers***

What are HTTP Security Headers?

- Response headers that the server responds with to instruct the browser what security rules to enforce when it handles your website’s content.
- Key value pairs
- In general, the more security headers you opt-in to sending, the more secure your website is.
- Most security headers come with multiple options you can configure to tweak the behavior to what you want.



```
facebook.com
www.facebook.com
MQadQzJY5-0.css
B9wZdYhoLBh.js
CZDXNrhcZNR.js
26UbA3YYKtW.js
lWpoemjeclb.css
o1ucC237c08.js
zm0ClAY-gWw.js
iXjARbyUKEH.css
jlLuFtmbjCq.css
IV5UITSC1cY.css
fOTwljApvzr.css
hKcmnd-vLZZ.js
ongZGhrl48M.css
ewdB3k4d_m3.css
R-Vbjiv5u5v.js
nzraUlbnMyJ.js
u2_4BeXClEg.js
495 / 536 requests | 5.9 MB / 6.0 ...
```

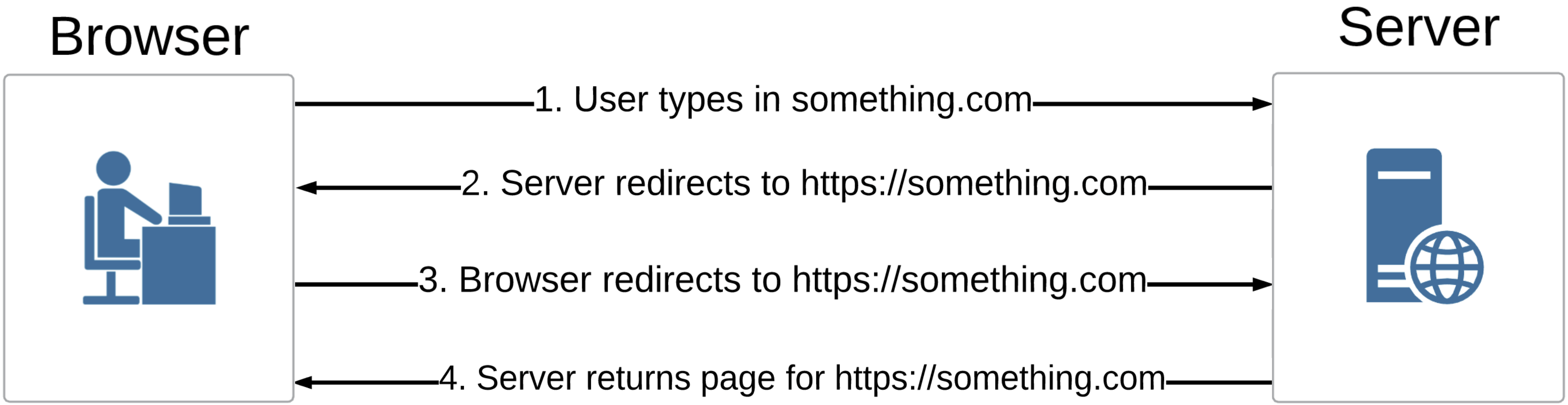
```
cache-control: private, no-cache, no-store, must-revalidate
content-encoding: br
content-security-policy: default-src * data: blob:;script-src *.facebook.com *.fbcdn.net *.facebook.net *.google-analytics.com *.virtualearth.net *.google.com 127.0.0.1:* *.spotilocal.com:* 'unsafe-inline' 'unsafe-eval' *.atla
olutions.com blob: data: 'self';style-src data: blob: 'unsafe-inline' *;connect-src *.facebook.com facebook.com *.fbcdn.net *.facebook.net *.spotilocal.com:* wss://*.facebook.com:* https://fb.scanandcleanlocal.com:* *.atla
olutions.com attachment.fbsbx.com ws://localhost:* blob: *.cdninstagram.com 'self' chrome-extension://boadgeojelhgndaghljhdicfkmllpafd chrome-extension://dliochdbjfkdbacpmhlcpmleaejidimm;
content-type: text/html; charset="utf-8"
date: Thu, 15 Nov 2018 21:15:41 GMT
expect-ct: max-age=86400, report-uri="http://reports.fb.com/expectct/"
expires: Sat, 01 Jan 2000 00:00:00 GMT
pragma: no-cache
status: 200
strict-transport-security: max-age=15552000; preload
vary: Accept-Encoding
x-content-type-options: nosniff
x-fb-debug: KkqYDI+lC5qzp2+H01/yAWcblI2/t3H/DmM51X5rJVZvcB+AlN9XNFaDftUb35k0Tnn4dCUT/bAizUfnv/E0yg==
x-frame-options: DENY
x-xss-protection: 0
```

scottsauber

HTTP Strict Transport Security (HSTS)

- What is it?
  - It allows websites to tell web browsers to only request this site over HTTPS, not over HTTP.
- Why should I care?
  - Prevents some classes of man-in-the-middle (MITM) attacks.

Without HSTS



Browser

Server

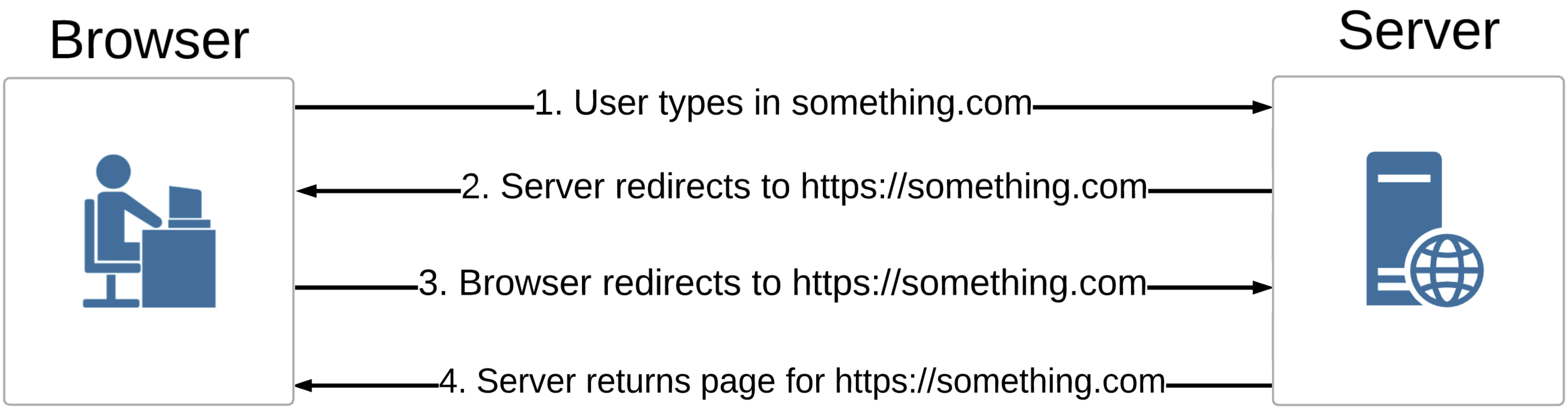
1. User types in something.com

2. Server redirects to https://something.com

3. Browser redirects to https://something.com

4. Server returns page for https://something.com

What’s the issue?



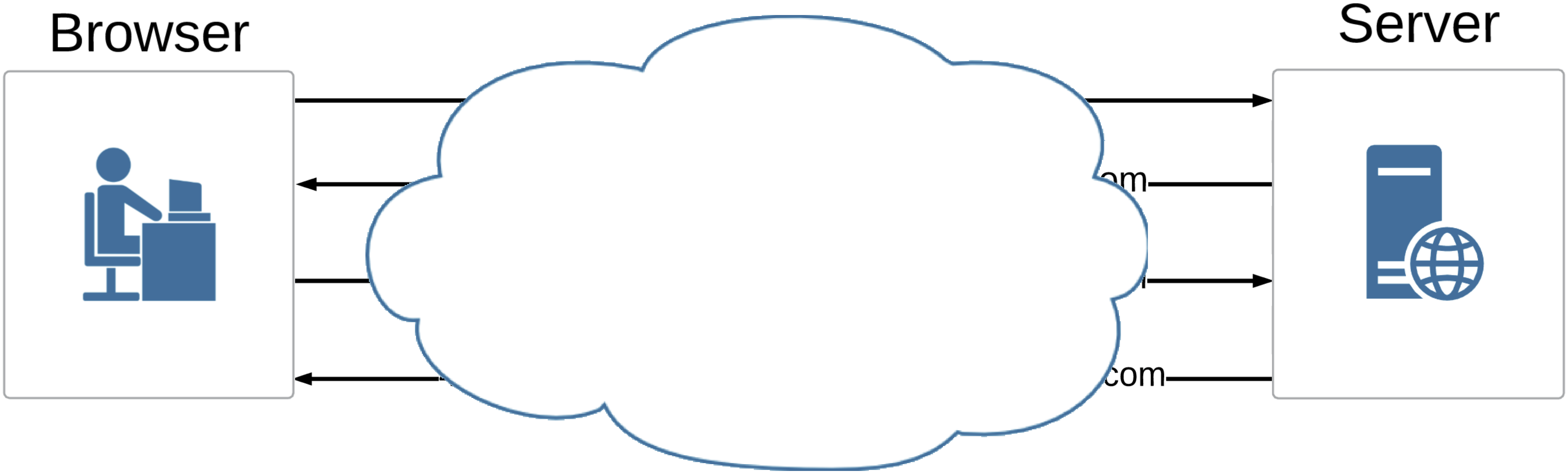
1. User types in something.com

2. Server redirects to https://something.com

3. Browser redirects to https://something.com

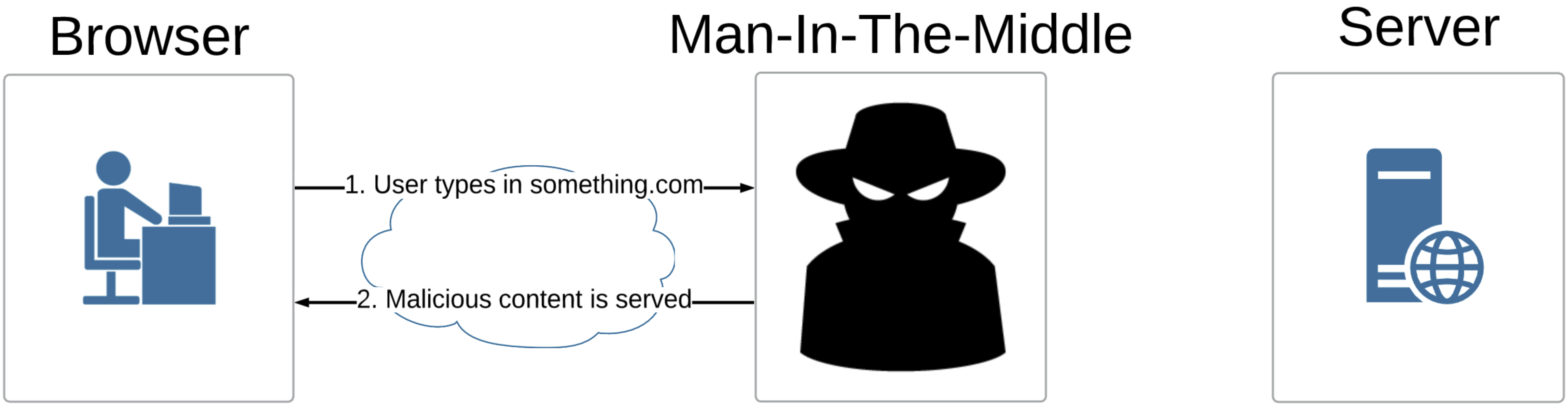
4. Server returns page for https://something.com

What’s the issue?



scottsauber

What can happen?



Browser

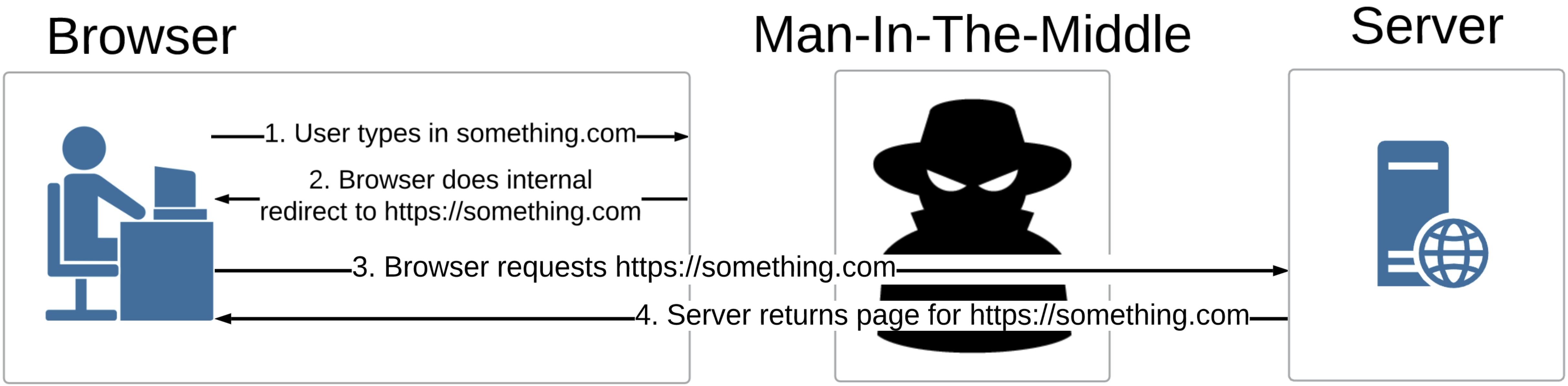
Man-In-The-Middle

Server

1. User types in something.com

2. Malicious content is served

With HTTS



Browser

Man-In-The-Middle

Server

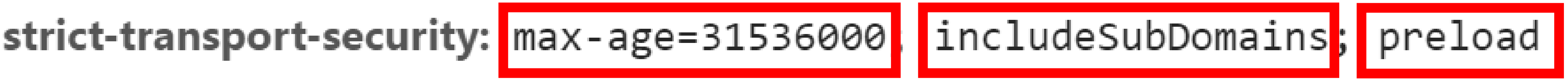
1. User types in something.com

2. Browser does internal redirect to https://something.com

3. Browser requests https://something.com

4. Server returns page for https://something.com

HSTS Options

Example: 

```
strict-transport-security: max-age=31536000; includeSubDomains; preload
```

- max-age
  - The number of seconds the browser should enforce HSTS. 31,536,000 (1 year) is really common. Adds your site to its internal list for this # of seconds.
- includeSubDomains
  - Apply the HSTS policy to all subdomains.
- preload
  - Instructs the browser to be on the preload list… more on that in the next slide.
- max-age is required. The other two are not.

HSTS Preload List

- List maintained by Google, but used by all browsers.
- If you **ARE NOT** on the list, then the first HTTP request will 301 and opens up for chance of MITM
- If you **ARE** on this list, then the HTTP request will 307 internal redirect, not 301, even if you've never visited the site before
- Guarantees no chance of basic MITM attack.
- Submit your domain to the list here: https://hstspreload.org/
- Add the preload option to your header to confirm your submission.

HSTS Demo

HSTS Gotchas

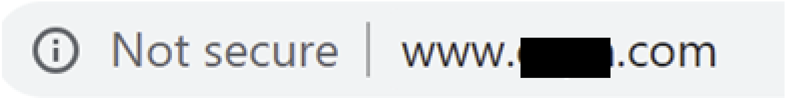
- You probably don’t want this running when running locally on localhost… unless every website you run locally is HTTPS
- HTTP and HTTPS often listen on different ports like localhost:5000 for HTTP and localhost:5001 for HTTPS.
  - If running for localhost:5000 it will redirect to https://localhost:5000 which will not bind

HSTS Impact of Retrofitting on Existing App

- Is everything really HTTPS?
- Subdomains
- If you’re planning on going from HTTPS to HTTP in the future for some reason
  - IDK why though

Quick word on HTTPS

- A good idea even if your site is internal
- Network topology may change
- Perception to users thanks to Chrome



ⓘ Not secure | www.[redacted].com

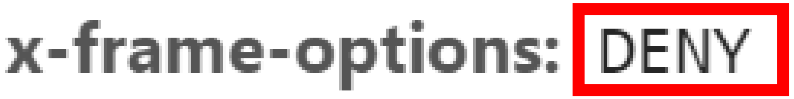
HSTS Questions

X-Frame-Options (XFO)

- What is it?
  - Used to tell a browser whether or not a page should be rendered in a frame or iframe.
- Why should I care?
  - Prevents click-jacking attacks.

X-Frame-Options (XFO) Options

Example:



x-frame-options: DENY

- Directives to choose from
  - DENY
    - Prevents any domain from framing your page. This is the **most secure.**
  - SAMEORIGIN
    - Only allows framing from the same domain.
  - ALLOW-FROM https://site1.com
    - Let’s you specify a single site that can frame your page.

XFO Demo

XFO Impact of Retrofitting to Existing App

- Do you know which sites should be iframing your app?
- I imagine most could just do DENY or at least SAMEORIGIN

XFO Questions

Cross-Site Scripting (XSS)

- What is it?
  - A vulnerability in a trusted website where malicious scripts can be injected.
  - XSS can be used to harvest cookies, tokens, etc. since the script that is loaded appears to be legit.

  - Often it comes from input from the user that is not validated or encoded and then re-displaying that to the user.
  - Examples:
    - Taking input from user, save it in a DB and others can see (Twitter, Facebook, etc.)
    - “Contact Us” or “Feedback” form on your page
      - Can you put in <script>//something malicious here</script> and does it get loaded by your email client?

XSS Demo

XSS Final Note

- Most modern frameworks help you out here.
- ASP.NET Core for instance, I have to call Html.Raw() since it encodes by default.
- React escapes non-props characters by default

XSS Questions

Cross-Site Scripting (XSS)

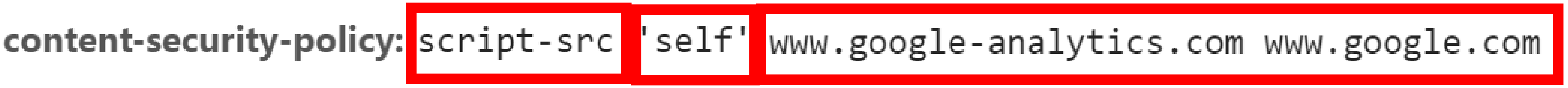
- Can be prevented with Content-Security-Policy (CSP)
  - Among other attacks not just XSS
- Old X-XSS-Protection security header is no longer honored by any major browser
  - Edge in 2018
  - Chrome in 2019

Content Security Policy (CSP)

- What is it?
    - Gives the browser an allowlist of sources to load static resources like JS, CSS, images, etc. from. This allowlist can specify how the resource is loaded (i.e. disabling inline scripts) and where the resource can be loaded from.
- Why should I care?
    - It can reduce or even eliminate the ability for XSS to occur.
    - Also limits your attack surface of other kinds of attacks (more later).

Content Security Policy (CSP) Options

Example:



```
content-security-policy: script-src 'self' www.google-analytics.com www.google.com
```

- script-src = the content type you are configuring
- self = the domain the page is being served on
- The rest are other domains that are allowed to load scripts from
- Other values:
  - unsafe-inline would mean allowing <script> tags or inline event handlers like <button onclick=“clickEvent”>
  - none means block any use of this content type
- report-uri = where to send JSON payload with violation information

Content Security Policy (CSP) Options

- In general, the more you allow, the greater your XSS risk.
- Not allowing inline scripts is one of the biggest wins if you can manage it.

Content Security Policy (CSP) Options

- There are other ones just like script-src that behave similarly such as:
  - style-src
  - media-src
  - frame-src
  - font-src
  - And more
- All take in domains to allow
- unsafe-inline also works with styles
- none works with all
  - i.e. if you want no one to frame your content

CSP
Demo

CSP Impacting of Retrofitting to Existing App

- **HUGE**
- This is an allowlist
- You **must know what your app is doing** (inline scripts/styles or not), where it's loading from (CDN's, other sources, or not), etc.
- Configuring this wrong will break your app.
- Compromise
  - Set to report only (via Content-Security-Policy-Report-Only instead of Content-Security-Policy), collect data and what your app does, and tweak CSP to that accordingly after a certain period of time.
  - Start converting inline scripts and the like.

Content Security Policy (CSP)

- CSP can override the need for other headers
- frame-ancestors ‘none’ means no one can embed the page in a frame/iframe.
  - This eliminates the need for X-Frame-Options: DENY
- However, auditors probably still want to see it

CSP Questions

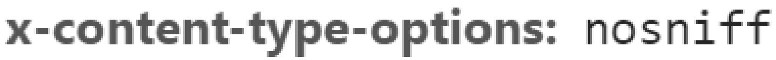
Browser Sniffing Protection (X-Content-Type-Options)

- What is it?
  - Tells a browser to not “sniff” the response and try and determine what’s in the response. Instead, look at the content-type header and render it according to that. So if it says it’s text/plain, render it as text/plain

- Why should I care?
  - Prevents unexpected execution from what the server thinks the response is.
  - Especially important if you take uploads from a user and re-display them.
  - Someone may upload a .txt file, but it’s really JavaScript and without this option set, the browser may execute the JavaScript.

Browser Sniffing Protection (X-Content-Type-Options)

Example:



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

- nosniff
  - Does not have the browser sniff the contents of the response to try and determine what to display
  - Instead, it just looks at the content-type header and renders it as that.

XCTO Impact of Retrofitting to Existing App

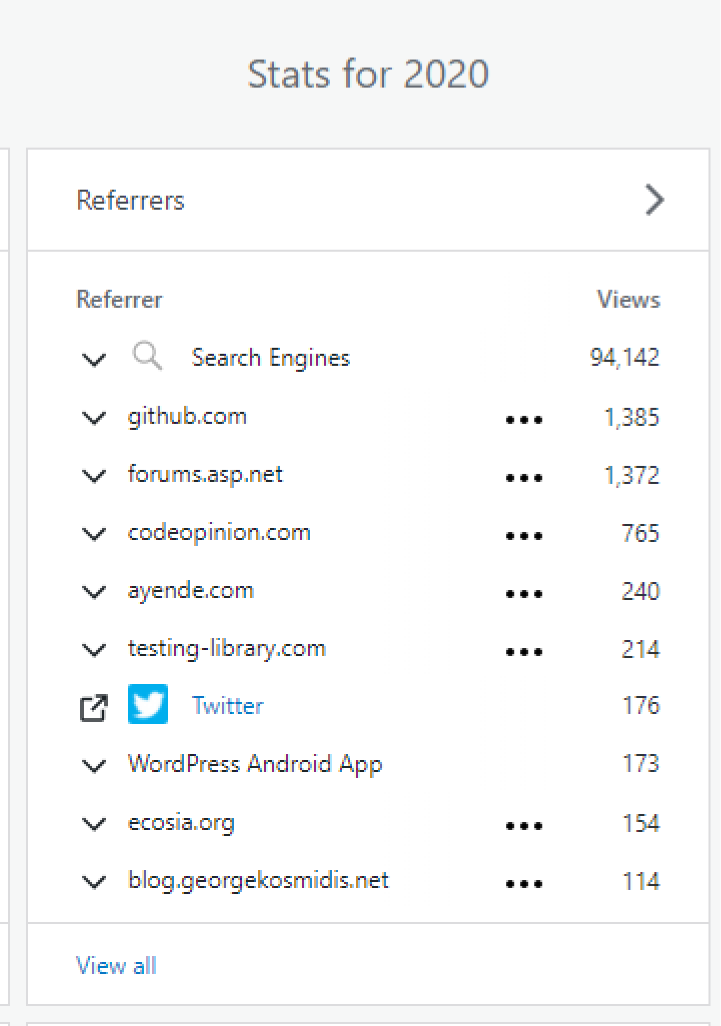
- Very minimal
- Note: most modern browsers will *not* sniff by default now.
- IE in compatibility view will still sniff
- Still shows up on audits

XCTO Questions

Referer Header background

- When a link is clicked, the browser will send the previous page’s URL in the Referer Request Header. Allows the server to do something with that data.
- Useful for tracking a user’s flow through an app
- Yes it’s misspelled
- Yes that’s actually how it shows up in the browser

I’ve seen this on my blog



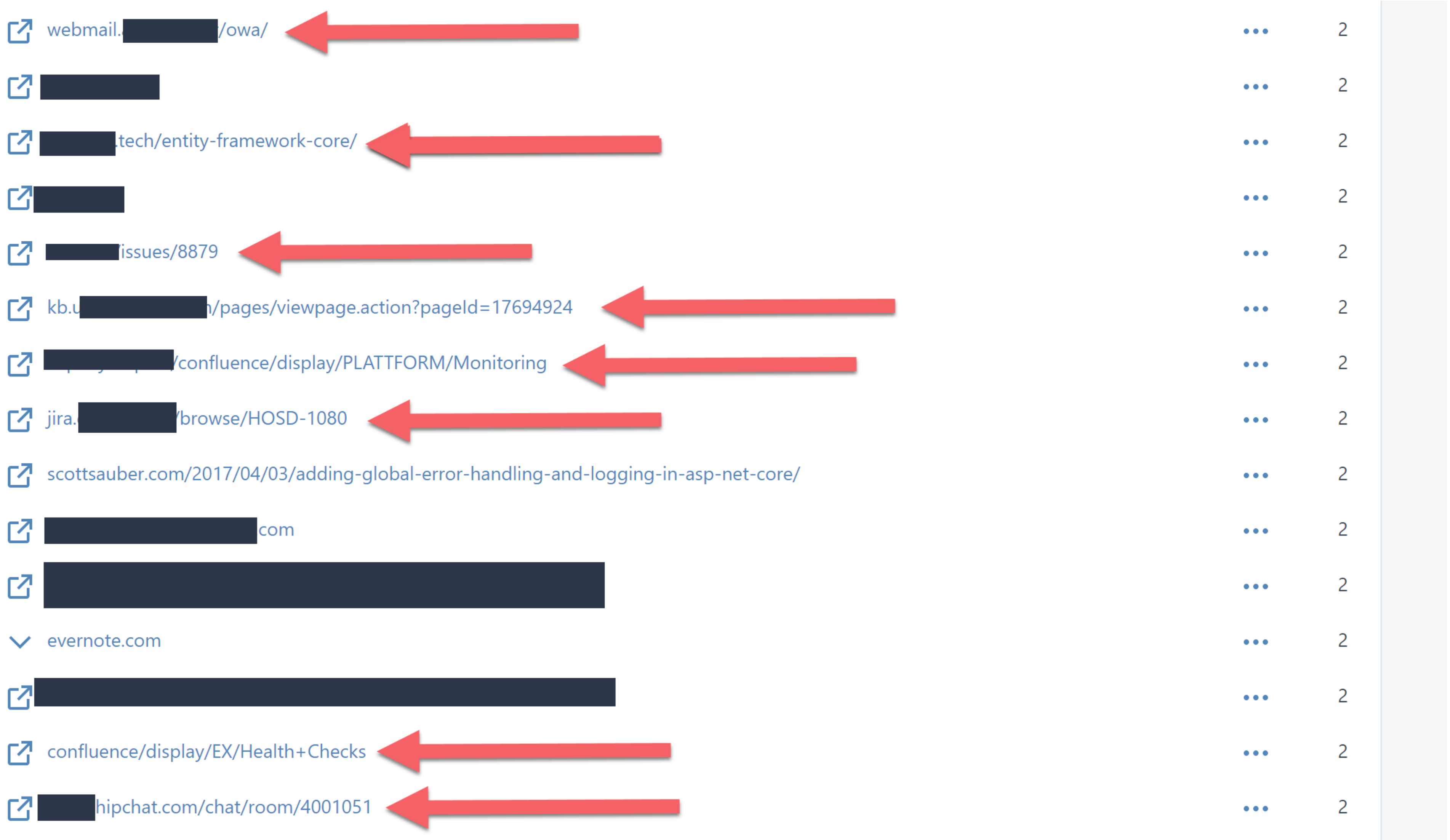
Stats for 2020

Referrers

Referrer	Views
Search Engines	94,142
github.com	1,385
forums.asp.net	1,372
codeopinion.com	765
ayende.com	240
testing-library.com	214
Twitter	176
WordPress Android App	173
ecosia.org	154
blog.georgekosmidis.net	114

View all

…and even JIRA/Confluence/OWA



webmail.[redacted]/owa/	•••	2
[redacted]	•••	2
[redacted]tech/entity-framework-core/	•••	2
[redacted]	•••	2
[redacted]issues/8879	•••	2
kb.[redacted]/pages/viewpage.action?pageId=17694924	•••	2
[redacted]/confluence/display/PLATTFORM/Monitoring	•••	2
jira.[redacted]/browse/HOSD-1080	•••	2
scottsauber.com/2017/04/03/adding-global-error-handling-and-logging-in-asp-net-core/	•••	2
[redacted]com	•••	2
[redacted]	•••	2
evernote.com	•••	2
[redacted]	•••	2
confluence/display/EX/Health+Checks	•••	2
[redacted]hipchat.com/chat/room/4001051	•••	2

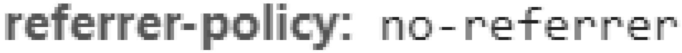
Referrer-Policy

- What is it?
  - Tells a browser what should be sent in the Referer header

- Why should I care?
  - It helps protect the identity of the source of a page’s visit.

Referrer-Policy

Example:



```
referrer-policy: no-referrer
```

- no-referrer
  - Referer header is omitted entirely. **Most secure.**
- origin
  - Only send the domain (i.e. sends example.com instead of example.com/index.html)
- same-origin
  - Only send when going to the same domain
- And more

RP Impact of Retrofitting to Existing App

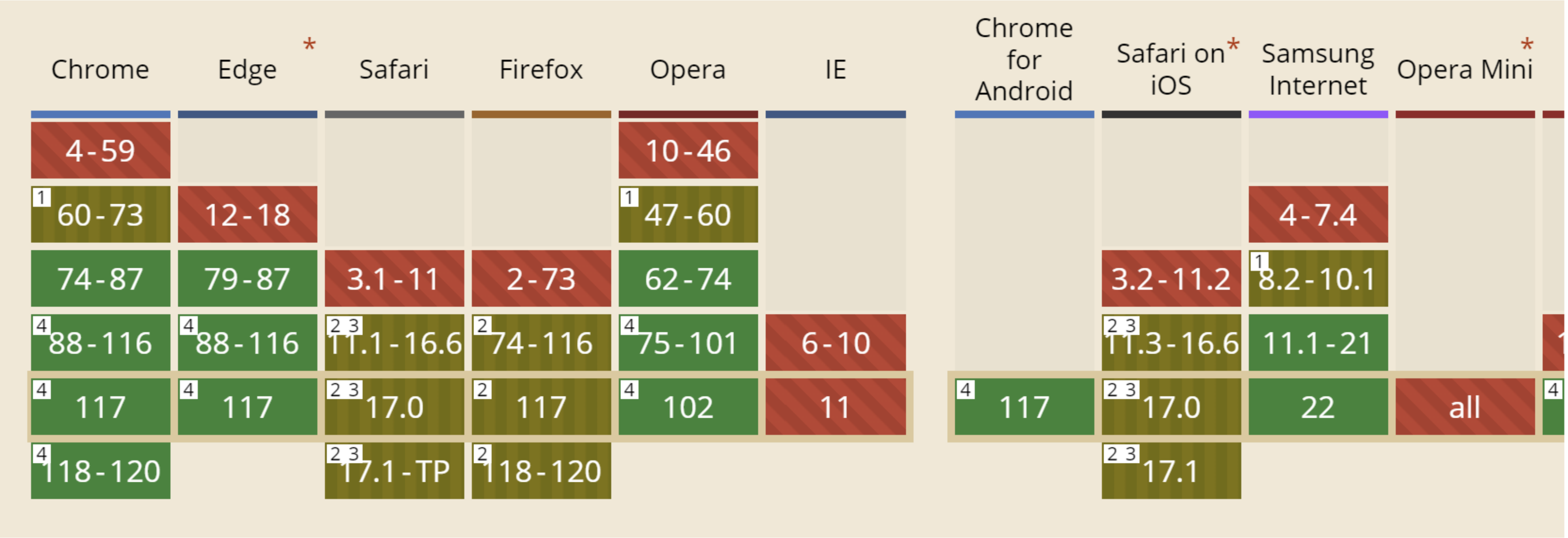
- Minimal with the right config

RP Questions

Feature-Policy (Working Draft)

- What is it?
    - Tells a browser to allow or deny the use of browser features, and allowing granularity of being able to specify specific domains
- Why should I care?
    - Allows you to restrict what your own app can do
        - In case of a XSS vulnerability
    - Allows you to restrict what 3rd party code can do
        - Block geolocation, camera, microphone, etc.
- Limited browser support – rename coming to “Permissions Policy”

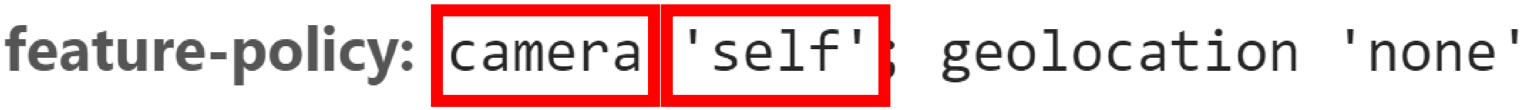
Feature-Policy Is Experimental



Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	
4-59				10-46						
[1] 60-73	12-18			[1] 47-60				4-7.4		
74-87	79-87	3.1-11	2-73	62-74			3.2-11.2	[1] 8.2-10.1		
[4] 88-116	[4] 88-116	[2 3] 11.1-16.6	[2] 74-116	[4] 75-101	6-10		[2 3] 11.3-16.6	11.1-21		
[4] 117	[4] 117	[2 3] 17.0	[2] 117	[4] 102	11	[4] 117	[2 3] 17.0	22	all	[4]
[4] 118-120		[2 3] 17.1-TP	[2] 118-120				[2 3] 17.1			

Feature-Policy

Example:



feature-policy: `camera 'self'; geolocation 'none'`

- The feature you are locking down
  - camera, geolocation, microphone, payment, autoplay, etc.
- The allow list of who can use this feature
  - *
  - self
  - none
  - https://example.com

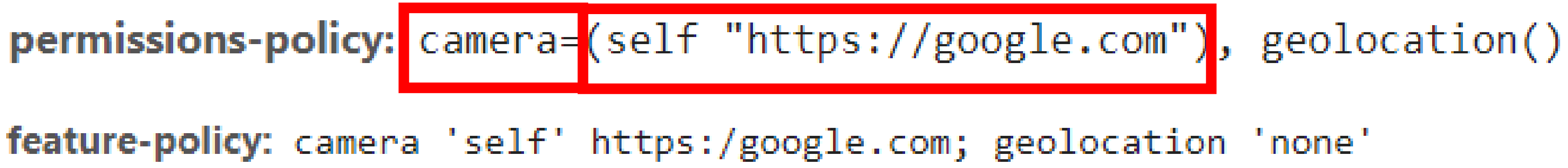
FP
Demo

FP Impact of Retrofitting to Existing App

- Pretty big
- Know what your site is doing

Permissions-Policy

Example:

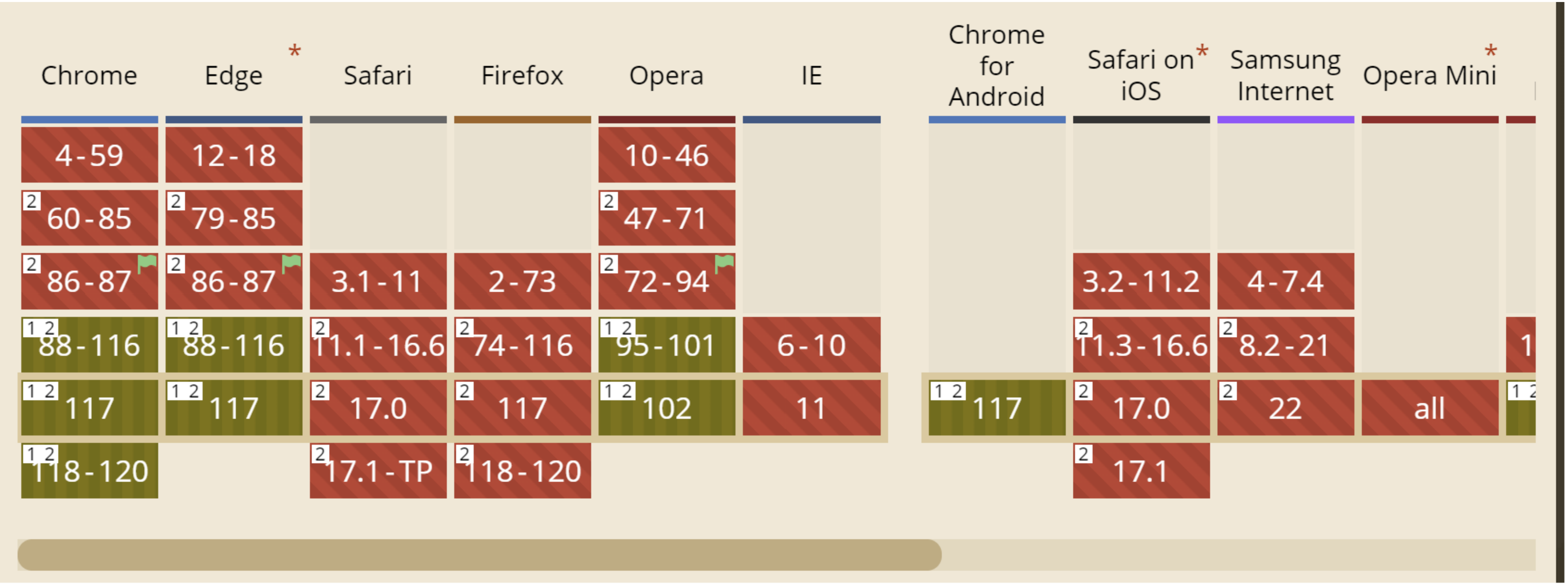


```
permissions-policy: camera=(self "https://google.com"), geolocation()

feature-policy: camera 'self' https:/google.com; geolocation 'none'
```

- Same idea as Feature-Policy but slightly different syntax
- The feature you are locking down
- The allow list of who can use this feature
- PP will (likely) replace FP, but it has almost zero support today unlike FP

Permissions-Policy is a Working Draft



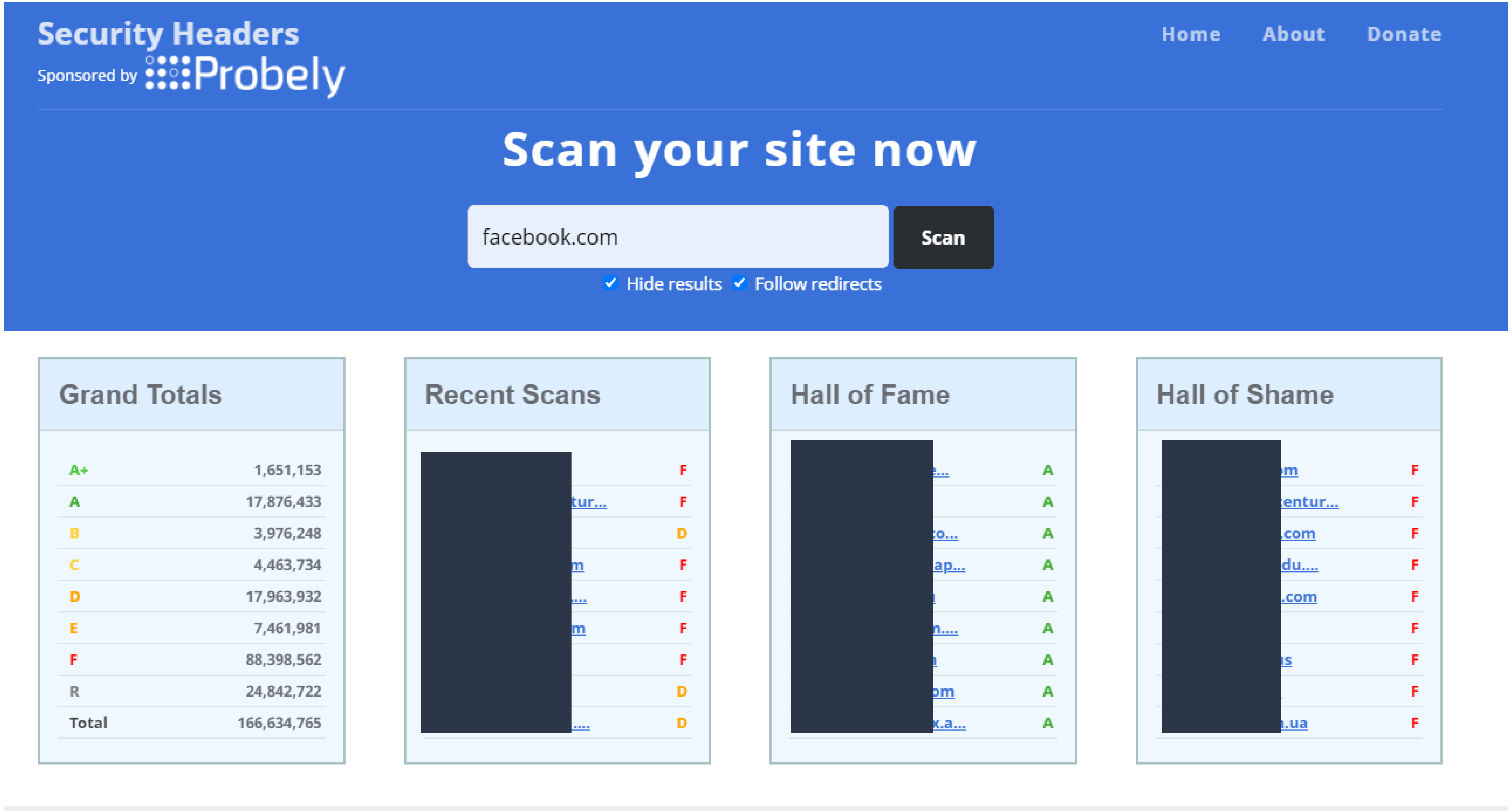
Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	
4-59	12-18			10-46						
[2] 60-85	[2] 79-85			[2] 47-71						
[2] 86-87 ⚑	[2] 86-87 ⚑	3.1-11	2-73	[2] 72-94 ⚑			3.2-11.2	4-7.4		
[1] [2] 88-116	[1] [2] 88-116	[2] 11.1-16.6	[2] 74-116	[1] [2] 95-101	6-10		[2] 11.3-16.6	[2] 8.2-21		1
[1] [2] 117	[1] [2] 117	[2] 17.0	[2] 117	[1] [2] 102	11	[1] [2] 117	[2] 17.0	[2] 22	all	[1] [2]
[1] [2] 118-120		[2] 17.1-TP	[2] 118-120				[2] 17.1			

FP/PP Questions

How do I test my website?

- https://securityheaders.com
- Run by security expert Scott Helme

SecurityHeaders.com



Scan your site now

facebook.com

☑ Hide results ☑ Follow redirects

Grand Totals

A+	1,651,153
A	17,876,433
B	3,976,248
C	4,463,734
D	17,963,932
E	7,461,981
F	88,398,562
R	24,842,722
Total	166,634,765

Recent Scans

[redacted]	F
[redacted]	F
[redacted]	D
[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	D
[redacted]	D

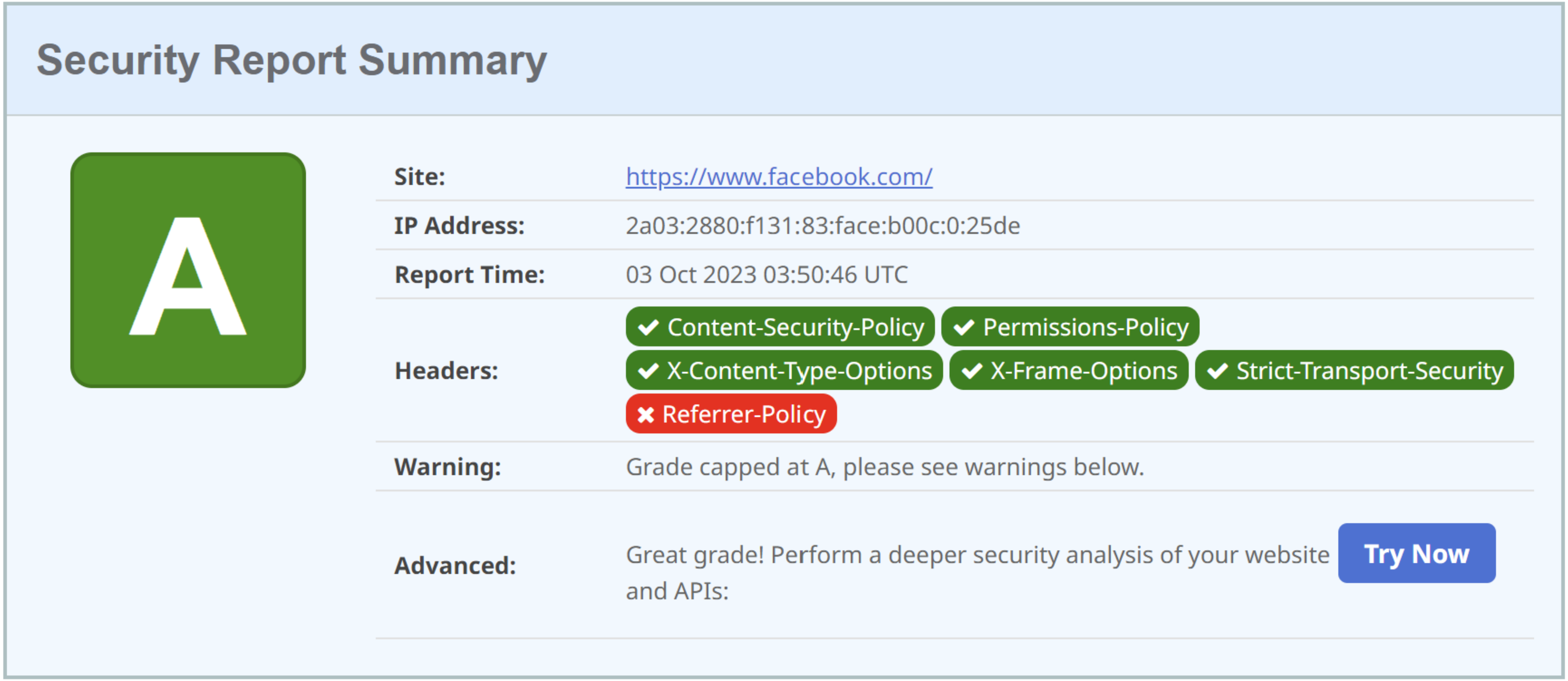
Hall of Fame

[redacted]	A
[redacted]	A
[redacted]	A
[redacted]	A
[redacted]	A
[redacted]	A
[redacted]	A
[redacted]	A
[redacted]	A

Hall of Shame

[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	F
[redacted]	F

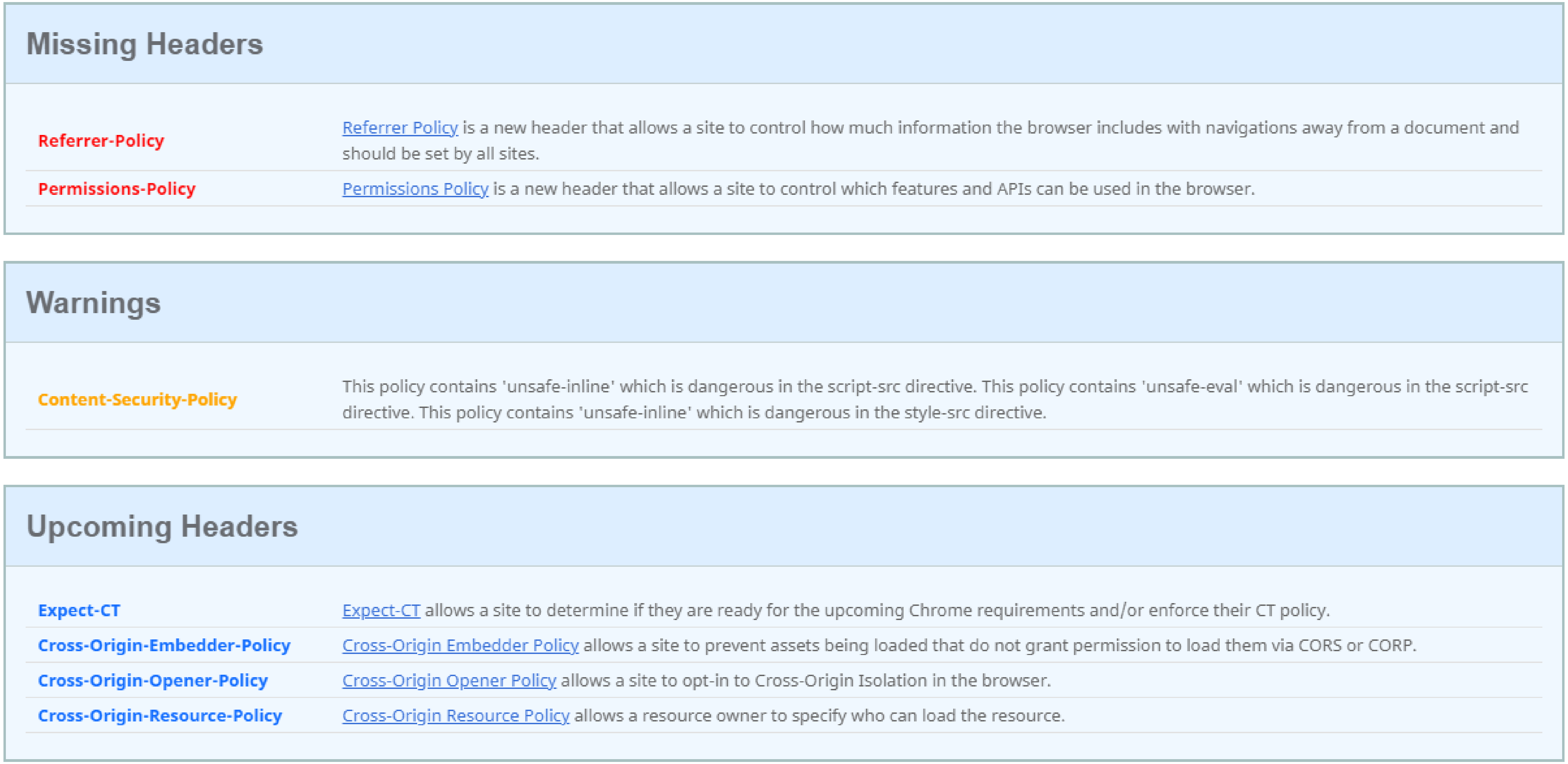
SecurityHeaders.com



Security Report Summary

Site:	https://www.facebook.com/
IP Address:	2a03:2880:f131:83:face:b00c:0:25de
Report Time:	03 Oct 2023 03:50:46 UTC
Headers:	✓ Content-Security-Policy ✓ Permissions-Policy ✓ X-Content-Type-Options ✓ X-Frame-Options ✓ Strict-Transport-Security ✗ Referrer-Policy
Warning:	Grade capped at A, please see warnings below.
Advanced:	Great grade! Perform a deeper security analysis of your website and APIs: Try Now

SecurityHeaders.com



Missing Headers

Referrer-Policy	Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
Permissions-Policy	Permissions Policy is a new header that allows a site to control which features and APIs can be used in the browser.

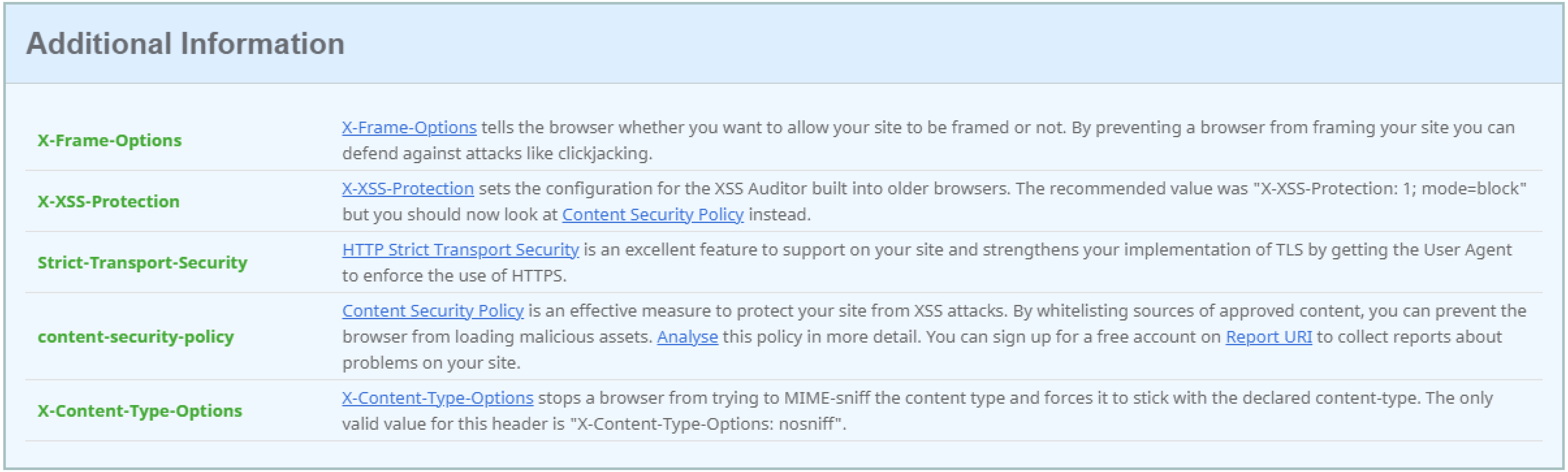
Warnings

Content-Security-Policy	This policy contains 'unsafe-inline' which is dangerous in the script-src directive. This policy contains 'unsafe-eval' which is dangerous in the script-src directive. This policy contains 'unsafe-inline' which is dangerous in the style-src directive.

Upcoming Headers

Expect-CT	Expect-CT allows a site to determine if they are ready for the upcoming Chrome requirements and/or enforce their CT policy.
Cross-Origin-Embedder-Policy	Cross-Origin Embedder Policy allows a site to prevent assets being loaded that do not grant permission to load them via CORS or CORP.
Cross-Origin-Opener-Policy	Cross-Origin Opener Policy allows a site to opt-in to Cross-Origin Isolation in the browser.
Cross-Origin-Resource-Policy	Cross-Origin Resource Policy allows a resource owner to specify who can load the resource.

SecurityHeaders.com



Additional Information

X-Frame-Options	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking.
X-XSS-Protection	X-XSS-Protection sets the configuration for the XSS Auditor built into older browsers. The recommended value was "X-XSS-Protection: 1; mode=block" but you should now look at Content Security Policy instead.
Strict-Transport-Security	HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS.
content-security-policy	Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets. Analyse this policy in more detail. You can sign up for a free account on Report URI to collect reports about problems on your site.
X-Content-Type-Options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".

Note

- If you’re using a WAF (Cloudflare, Incapsula, etc.) they may be adding these for you.
- Personally, I’d rather let the app add them, avoid vendor-lock in, and get localhost running as close to prod as possible.
- Sometimes this is hard to do if doing JAM stack
  - Lambda@Edge

Takeaways

- HTTP Security Header Awareness
- At least one HTTP Header or option written down to look into at work
- There are more Security Headers out there and more coming
- SecurityHeaders.com
- The web is a scary place

scottsauber

Resources

- https://securityheaders.com/
- MDN: https://developer.mozilla.org/en-US/docs/Web/HTTP/
  - Http Security on the left
- Code from demos: https://github.com/scottsauber/talks
- Troy Hunt Pluralsight on Security Headers
- This slide deck is intentionally left detailed

Questions?

Thanks!

scottsauber