

# HTTP Security Headers You Need To Know

Scott Sauber



# Audience

- Anyone with a website/webapp



# Agenda

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

# Purpose

- 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 the office

# Who am I?

- Software Consultant at [Lean TECHniques](#)
- Primarily C# and JS Developer
- 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.

The screenshot shows the Chrome DevTools Network tab. The top toolbar includes 'View: Group by frame', 'Preserve log', 'Disable cache', and 'Offline Online'. The network list on the left shows a request to 'www.facebook.com' selected. The response headers are expanded, showing various headers such as 'cache-control', 'content-encoding', 'content-security-policy', 'content-type', 'date', 'expect-ct', 'expires', 'pragma', 'status', 'strict-transport-security', 'vary', 'x-content-type-options', 'x-fb-debug', 'x-frame-options', and 'x-xss-protection'. Red arrows point to the 'www.facebook.com' entry in the network list (labeled '2'), the 'Response Headers' section (labeled '3'), and the 'x-fb-debug' header value.

Network tab view showing response headers for a request to `www.facebook.com`.

**Response Headers:**

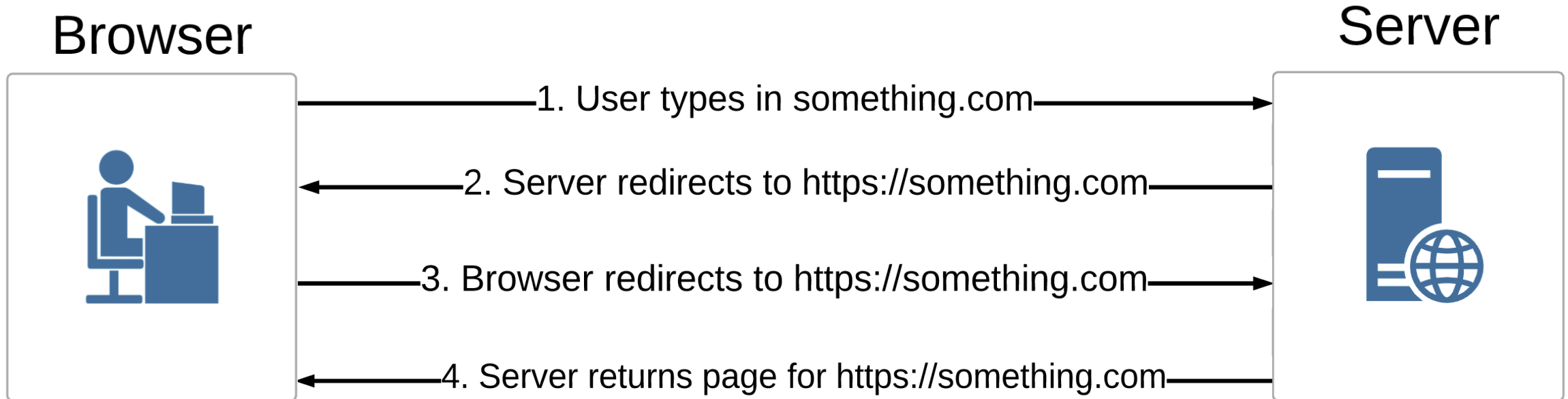
- `cache-control:` private, no-cache, no-store, must-revalidate
- `content-encoding:` br
- `content-security-policy:` default-src \* data: blob:;script-src \*.facebook.com \*.fbcdn.net \*.facebook.net \*.google-analytics.com \*.virtualearth.net \*.google.com 127.0.0.1:\* \*.spotilocal.com:\* 'unsafe-inline' 'unsafe-eval' \*.atlaolutions.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:\* \*.atlaolutions.com attachment.fbsbx.com ws://localhost:\* blob: \*.cdninstagram.com 'self' chrome-extension://boadgeojelhgndaghljhdicfkmllpafd chrome-extension://dliochdbjfkdbacpmhlcpmlaejdim;
- `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+1C5qzp2+H01/yAWcb1I2/t3H/DmM51X5rJVZvcB+A1N9XNFaDftUb35k0Tnn4dCUT/bAizUfnv/E0yg==
- `x-frame-options:` DENY
- `x-xss-protection:` 0



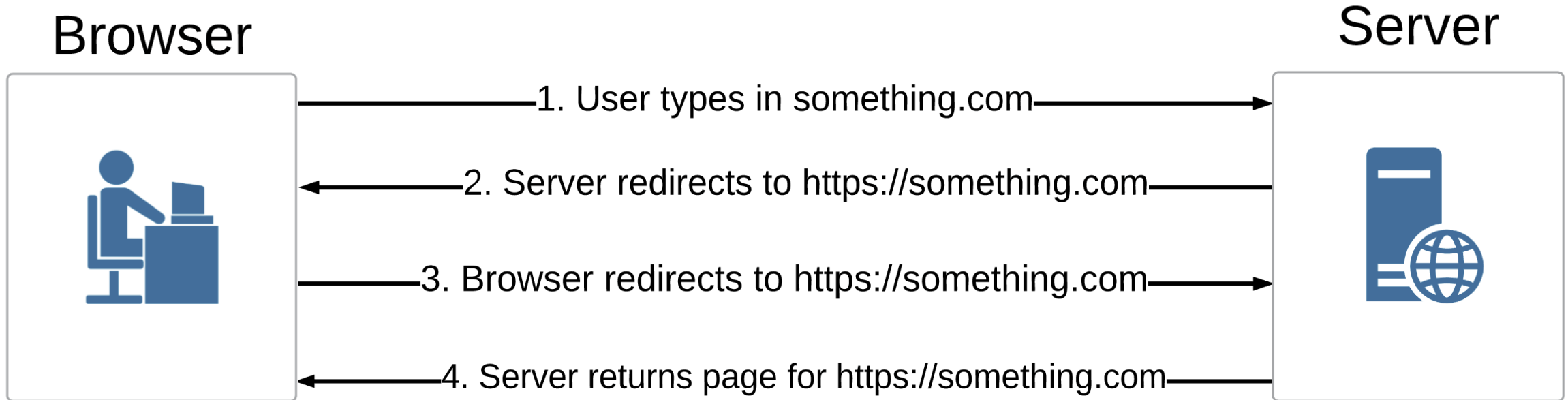
# HTTP Strict Transport Security (HSTS)

- 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.

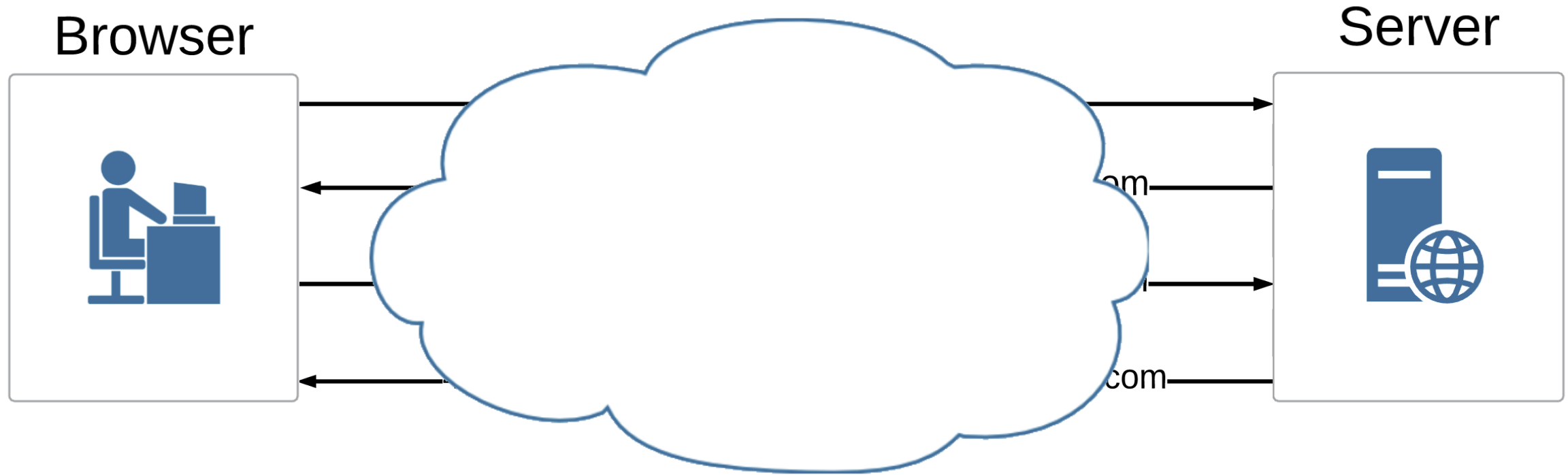
# Without HSTS



# What's the issue?



# What's the issue?

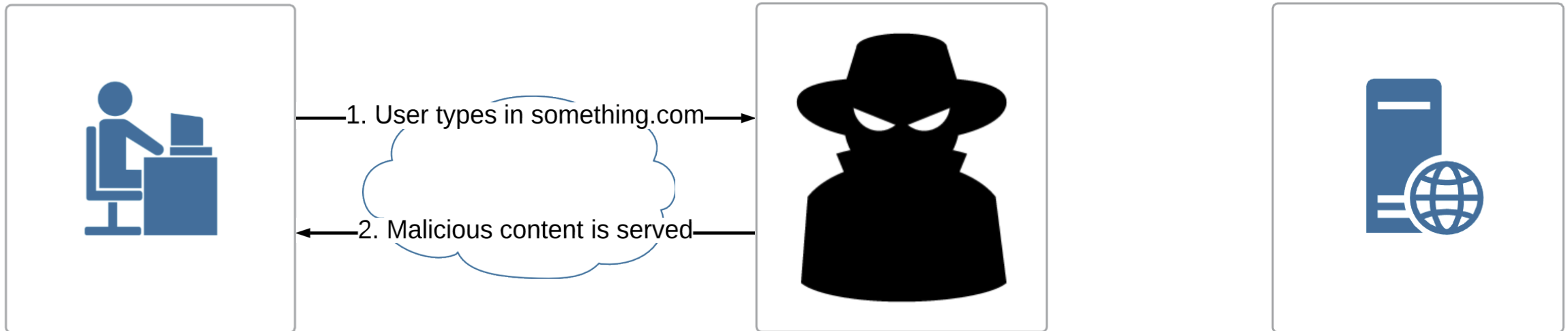


# What can happen?

Browser

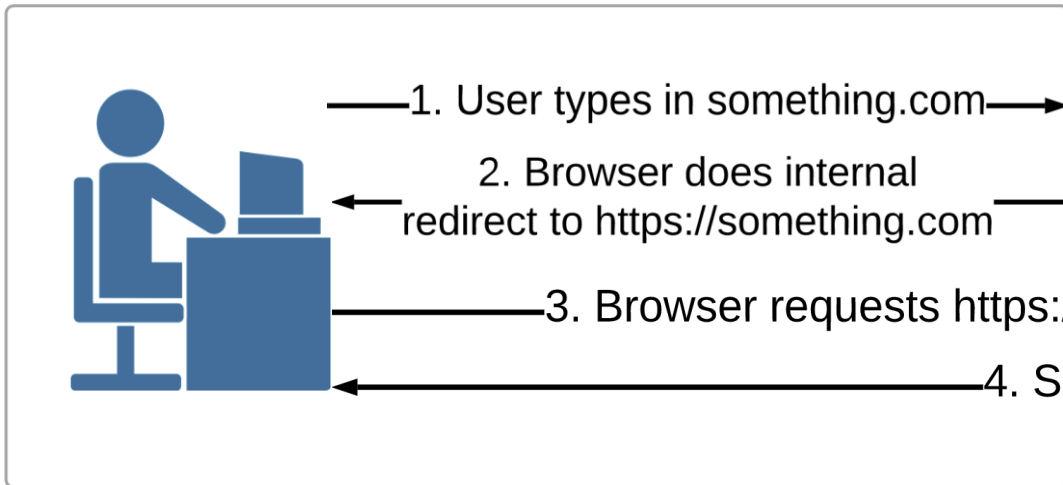
Man-In-The-Middle

Server



# With HSTS

## Browser



## Man-In-The-Middle



## Server



# 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.
- Example above is the most secure form.

# 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 first HTTP request will 307 internal redirect, not 301.
- 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.



# HTTP Strict Transport Security (HSTS)

- Demo

- <https://nohstssecurityheaderstalk.azurewebsites.net/> – no HSTS
- <https://hstssecurityheaderstalk.azurewebsites.net/> – has HSTS

# 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

# 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.espn.com](http://www.espn.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 http://site1.com
    - Let's you specify a single site that can frame your page.

# X-Frame-Options (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 database, and re-displaying it on a “Review” page.
  - Passing data in from URL and re-displaying it.
  - “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?

# Cross-Site Scripting (XSS)

- Demo

# XSS Final Note

- Most modern frameworks/browsers 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
- Chrome I had to tell to let XSS happen via `X-XSS-Protection: 0`

XSS Questions before we talk about how to prevent it?

# Cross-Site Scripting Protection (X-XSS)

- What is it?
  - Tells a browser to protect a page if the browser detects cross-site scripting with its built-in XSS detection algorithm.
- Why should I care?
  - It helps prevent Cross-Site Scripting

# Cross-Site Scripting Protection (X-XSS)

Example: `x-xss-protection: 1; mode=block; report=https://scotthelme.report-uri.com/r/d/xss/enforce`

- On or off
  - 0 = No XSS filtering
  - 1 = Enables XSS filtering and the browser will remove the unsafe part and continue rendering the page
- Mode
  - block = it will stop the page from rendering instead of removing the unsafe part.
- Report
  - URL to send a JSON report to describing the violation.



# Cross-Site Scripting Protection (X-XSS)

- Demo

# X-XSS Impact of Retrofitting to Existing app

- Fairly minimal, unless allowing arbitrary JS is in your app's wheelhouse.

# Cross-Site Scripting Protection Questions?

# Cross-Site Scripting (XSS)

- First layer of defense: Cross-Site Scripting Protection
- Second layer of defense: Content-Security-Policy (CSP)
  - Among other attacks not just XSS

# Content Security Policy (CSP)

- What is it?
  - Gives the browser a whitelist of sources to load static resources like JS, CSS, images, etc. from. This whitelist can include multiple domains as well as how the resource is loaded (i.e. disabling inline scripts).
- 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 whitelisting
- `self` = the domain the page is being served on
- The rest are other domains that are whitelisted 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



# Content Security Policy (CSP)

- Demo
- Harlem Shake

# CSP Impacting of Retrofitting to Existing App

- **HUGE**

- This is a whitelist
- 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

# Content Security Policy 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

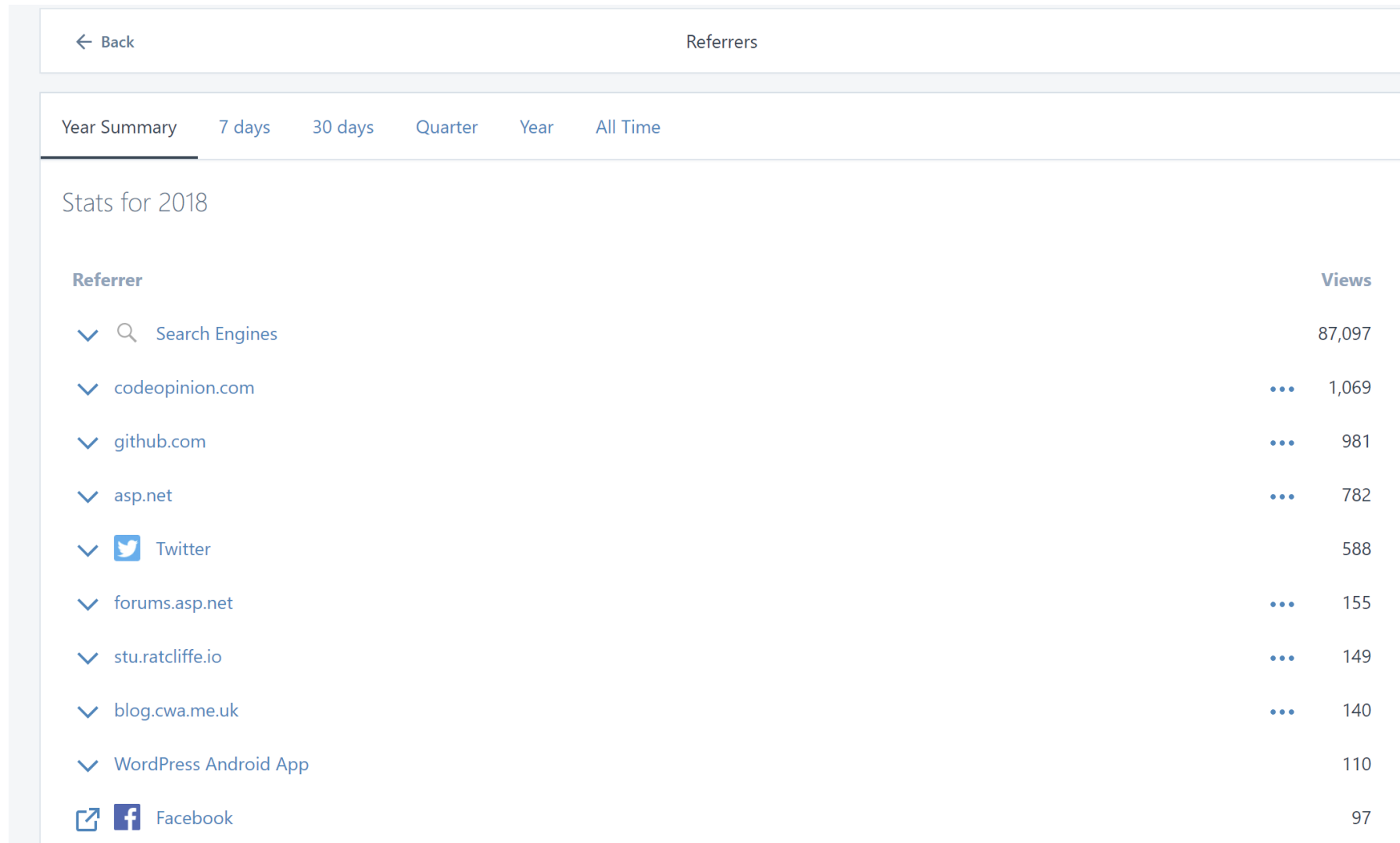
# Browser Sniffing Protection Questions?



# 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



The screenshot shows a web analytics dashboard titled "Referrers". At the top left, there is a "Back" button. Below the title, there are navigation tabs for "Year Summary", "7 days", "30 days", "Quarter", "Year", and "All Time". The main content area is titled "Stats for 2018" and displays a table of referrers. The table has two columns: "Referrer" and "Views". The referrers are listed in descending order of views. Each row includes a dropdown arrow, an icon, the referrer name, and the view count. Some rows also have a three-dot menu icon.

Referrer	Views
Search Engines	87,097
codeopinion.com	1,069
github.com	981
asp.net	782
Twitter	588
forums.asp.net	155
stu.ratcliffe.io	149
blog.cwa.me.uk	140
WordPress Android App	110
Facebook	97



# 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
  - Referrer 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

# Referrer-Policy Questions?

# Feature-Policy Is Coming

- 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
  - Think – 3<sup>rd</sup> party code you embed.
- 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 3<sup>rd</sup> party code can do
    - Block geolocation, camera, microphone, etc.
- Limited support in Chrome and Firefox now



# 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>

# Feature-Policy Demo

- Demo

# FP Impact of Retrofitting to Existing App

- Pretty big
- Know what your site is doing

# How do I test my website?

- <https://securityheaders.com>
- Run by security expert [Scott Helme](#)

# SecurityHeaders.com

Security Headers  
Sponsored by **netsparker**

[Home](#) [About](#)

## Scan your site now

Scan

Hide results  Follow redirects

### Grand Totals

A+	812,472
A	5,586,816
B	1,829,518
C	1,074,454
D	3,843,549
E	2,672,374
F	19,740,058
R	3,567,626
Total	39,126,867

### Recent Scans

xn	E
ne	F
er	F
ga	F
w	F
ac	E
ea	F
el	F
ch	F

### Hall of Fame

ww	A
ww	A
fat	A
nl	A
ww	A
an	A+
cyb	A
for	A+
pie	A

### Hall of Shame

ne	F
er	F
ga	F
w	F
oa	F
ea	F
el	F
ch	F
sa	F

# SecurityHeaders.com

## Scan your site now

 Hide results  Follow redirects

### Security Report Summary



**Site:** <https://www.facebook.com/>

**IP Address:** 2a03:2880:f131:83:face:b00c:0:25de

**Report Time:** 01 Jul 2019 03:13:49 UTC

**Headers:**  Strict-Transport-Security  Content-Security-Policy  X-Content-Type-Options  X-Frame-Options  
 X-XSS-Protection  Referrer-Policy  Feature-Policy

**Warning:** Grade capped at A, please see warnings below.

# 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.

### Feature-Policy

[Feature 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.

## 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.

# SecurityHeaders.com

## Additional Information

### 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. [Analyze](#) 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".

### 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 cross-site scripting filters built into most browsers. The best configuration is "X-XSS-Protection: 1; mode=block".



# 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



# 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/scottsauer/security-headers-talk>
- [Troy Hunt Pluralsight on Security Headers](#)
- Slides: <https://scottsauer.com>

Questions?

Thanks!