

LET'S BREAK STUFF

CONTENT SECURITY POLICIES

THINGS I DO

- ▶ Dungeon master for D&D campaigns
- ▶ Mentor & teach developers
- ▶ Speak at user groups and conferences
- ▶ **Break production sites with incorrectly configured content security policies**

OH GOOD, FINALLY WE'RE GETTING STARTED

A TALK IN 3 PARTS

XSS

CSP

BREAK
STUFF

THE SCARY STUFF

WHAT IS CROSS-SITE-SCRIPTING (XSS)?

- ▶ XSS enables an attacker to inject client-side scripts into **non-malicious** web pages viewed by other users
- ▶ It's (still) in the OWASP top 10
- ▶ In 2016 there was a 61% likelihood of a **browser-based** vulnerability being found in a web application
- ▶ Of those browser based vulnerabilities, **86% were found to be XSS related**
- ▶ **That was just over 52% of all web application vulnerabilities**
<https://www.edgescan.com/assets/docs/reports/2016-edgescan-stats-report.pdf>

I MEAN, IT'S JUST A JOKE VULNERABILITY, RIGHT?!

WHAT CAN BE DONE WITH XSS?

- ▶ Put pictures of ~~eats~~ dogs in web pages
- ▶ `alert('🐞');`
- ▶ Rickroll a user
- ▶ Twitter self-retweeting tweet
<https://www.youtube.com/watch?v=zv0kZKC6GAM>
- ▶ Samy worm
[https://en.wikipedia.org/wiki/Samy_\(computer_worm\)](https://en.wikipedia.org/wiki/Samy_(computer_worm))

WHAT CAN BE DONE WITH XSS?

- ▶ Make modifications to the DOM - replace a form action to point to your own script to capture credentials (logins, bank details, anything in a form really)
- ▶ Load in additional scripts, resources, styles, images etc (such as a crypto miner to make some \$\$\$)
- ▶ Access HTML5 APIs - webcam, microphone, geolocation
- ▶ Keylogging
- ▶ Phishing
- ▶ Steal cookies (and therefore steal session cookies)

WHAT CAN BE DONE WITH XSS?

KEVIN POULSEN SECURITY 03.28.08 08:00 PM

HACKERS ASSAULT EPILEPSY PATIENTS VIA COMPUTER

<https://www.wired.com/2008/03/hackers-assault-epilepsy-patients-via-computer/>

STORED XSS (AKA PERSISTENT OR TYPE I)

- ▶ Occurs when input is stored - generally in a server-side database, but not always
- ▶ This could also be within a HTML5 database, thus never being sent to the server at all
- ▶ who.is was a site Rickrolled by a TXT record in the DNS of a website **(yes, really)**

REFLECTED XSS (AKA NON-PERSISTENT OR TYPE II)

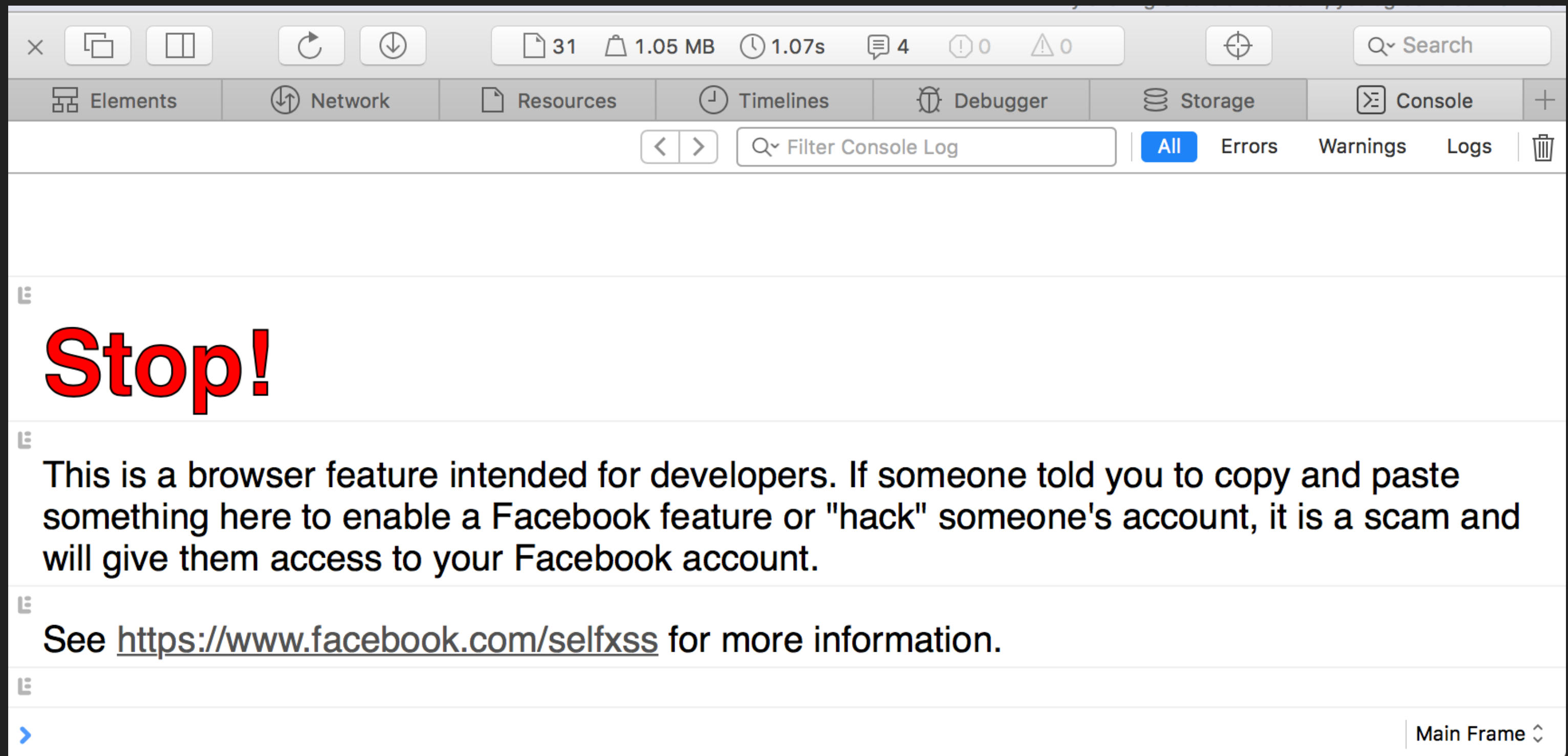
- ▶ Occurs when user input provided in the request is immediately returned - such as in an error message, search string etc
- ▶ Data is not stored, and in some instances, may not even reach the server (see the next type of XSS)

DOM-BASED XSS (AKA TYPE-0)

- ▶ The entire flow of the attack takes place within the browser
- ▶ For example, if JavaScript in the site takes input, and uses something like `document.write` based on that input, it can be vulnerable to a DOM-based XSS attack

SELF XSS

- ▶ Social-engineering form of XSS
- ▶ Requires the user to execute code in the browser
- ▶ Doing so via the console can't be protected by a lot of methods
- ▶ Not considered a 'true' XSS attack due to requiring the user to execute the code



A grayscale photograph of a person wearing a white lab coat with a skull logo on the sleeve, sitting at a desk and typing on a laptop. The desk is cluttered with various items, including a can, cables, and a sign that says "Nottingham Hackspace" and "Ask me at member". The background shows a workshop environment with shelves and equipment.

LET'S FIGHT BACK

WHAT IS A CSP?

HTTP RESPONSE HEADER TO HELP
REDUCE XSS RISKS

WHAT IS A CSP?

IT IS NOT A SILVER BULLET

WHAT IS A CSP?

IT IS AN EXTRA LAYER OF SECURITY

HOW DOES A CSP WORK?

DECLARES WHAT RESOURCES ARE
ALLOWED TO LOAD

IT CAN EVEN HELP WITH

BLOCKING THOSE PESKY CRYPTO-MINING
SCRIPTS THAT HAVE BEEN POPPING UP

BROWSER SUPPORT

Header	Chrome	FireFox	Safari	IE	Edge
Content-Security-Policy CSP Level 2	40+ Full January 2015	31+ <i>Partial</i> July 2014	10+	-	Edge 15 build 15002+
Content-Security-Policy CSP 1.0	25+	23+	7+	-	Edge 12 build 10240+
X-Content-Security-Policy Deprecated	-	4+	-	10+ <i>Limited</i>	12+ <i>Limited</i>
X-WebKit-CSP Deprecated	14+	-	6+	-	-

It's sort-of alright
Sorry IE users

WHAT CAN WE PROTECT?

- ▶ default-src
- ▶ script-src
- ▶ style-src
- ▶ img-src
- ▶ form-action
- ▶ upgrade-insecure-requests
- ▶ and so much more...

CONTENT-SECURITY-POLICY.COM

img-src *

ALLOWS ANY URL EXCEPT
DATA: BLOB: FILESYSTEM:
SCHEMES.

object-src 'none'

DON'T LOAD RESOURCES
FROM ANY SOURCE

style-src 'self'

ALLOW LOADING FROM
SAME SCHEME, HOST AND
PORT

script-src 'unsafe-inline'

ALLOWS USE OF INLINE
SOURCE ELEMENTS SUCH AS
STYLE ATTRIBUTE, ONCLICK, OR
SCRIPT TAG BODIES

DON'T USE
UNSAFE-
INLINE



```
script-src 'self' 'nonce-$RANDOM'
```

```
<script nonce="$RANDOM">...</script>
```



```
Content-Security-Policy: default-src 'none';  
script-src 'self' https://*.google.com 'nonce-  
random123'; style-src 'self'; img-src 'self';  
upgrade-insecure-requests; form-action 'self';
```

LEARN FROM MY MISTAKES

I BROKE PRODUCTION WITH A BAD CSP







DON'T DO WHAT I DID

REPORT-URI

WHEN A POLICY FAILURE OCCURS,
THE BROWSER SENDS A JSON
PAYLOAD TO THAT URL

```
{  
  "csp-report": {  
    "blocked-uri": "self",  
    "document-uri": "https://mysite.com",  
    "line-number": 1,  
    "original-policy": "script-src 'self'",  
    "script-sample": "try { for(var lastpass_iter=0; lastpass...",  
    "source-file": "https://mysite.com",  
    "violated-directive": "script-src 'self'"  
  }  
}
```

REPORT-URI.IO

Directive	Blocked URI	Raw	Count
<div>All ▾</div>	<div>blocked hostname</div> <div>blocked path</div>		<div>All ▾</div>
script-src	https://disqus.com/next/config.js	show/hide	1 
script-src	eval	show/hide	1 
script-src	https://c.disquscdn.com/next/embed/common.bundle.8acee1de90e869efdb244e45c7f66630.js	show/hide	1 
script-src	eval	show/hide	1 
script-src	eval	show/hide	1 
script-src	https://c.disquscdn.com/next/embed/lounge.bundle.9becee0326ce4d1840f8985f1dc0ce21.js	show/hide	1 

REPORT-ONLY

Content-Security-Policy-Report-Only:
[policy]; report-uri [URL];

TRIAL STUFF BEFORE ENFORCING

THERE WILL BE NOISE,
LOTS OF NOISE

DEMO TIME!
LET'S BREAK STUFF

TIPS

- ▶ Have an easy and quick way to disable the CSP in production if required
- ▶ Better yet, have a way to switch it from enforced to report only so you can get violations reported to help you debug
- ▶ Add the CSP at an application level if you need a nonce

NONCES

- ▶ Don't generate multiple nonces in the same request (but **do** generate a new nonce on each separate request)
- ▶ If using a templating engine (such as twig) - add the nonce as a global so it's available in every template by default
- ▶ Write a helper in your template engine to generate script tags with a nonce if it's available

MULTIPLE POLICIES

- ▶ They complicate things
- ▶ For a resource to be allowed, it must be allowed by all policies declared (problematic if an enforced policy)
- ▶ I tend to avoid them where possible on enforced policies
- ▶ But with report-only mode they can be very useful to deploy and test multiple policies at the same time (as nothing breaks for the user)

@SCOTT_HELME

HE KNOWS HIS STUFF!

@MR_GOODWIN

HE FIRST INTRODUCED ME
TO WHAT A CSP IS

LINKS & FURTHER READING

- ▶ [https://www.owasp.org/index.php/Cross-site Scripting \(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- ▶ <https://content-security-policy.com>
- ▶ <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>
- ▶ <https://report-uri.io>
- ▶ <https://scotthelme.co.uk/just-how-much-traffic-can-you-generate-using-csp/>
- ▶ <https://www.edgescan.com/assets/docs/reports/2016-edgescan-stats-report.pdf>
- ▶ <http://theharmonyguy.com/oldsite/2011/04/21/recent-facebook-xss-attacks-show-increasing-sophistication/>
- ▶ <https://github.com/Brunty/csp-demo>

THANK YOU

@BRUNTY

NOTI.ST/BRUNTY

MFYU.CO.UK