Care & Feeding of Programmers: Addressing App Sec Gaps using HTTP Headers

Sunny Wear

OWASP Tampa Chapter
December Meeting

About the Speaker

- Information Security Architect
  - Areas of expertise: Application, Network and Data Security Architecture
- Author – Secure Coding Field Manual available on Amazon

- Educator/Mentor/Coach/Consultant:
  - Secure Coding
  - Static Code Analysis
  - Manual Security Code Reviews
  - Secure Designs and Architecture Principles
  - Programmer understanding of Penetration Tests Results
- Contact:
  @SunnyWear

2013 celebrity photo hack

- Apple iCloud (hack occurring in 2013)
- Naked celebrity photos

Apple Knew About iCloud Flaw 6 Months Before 'The Fappening' Hit Celebrity Photos, Report Claims

By Kukil Bora @KukilBora on September 25 2014 8:09 AM EDT

On Sept. 1, hackers posted nude photos of celebrities, including those of Jennifer Lawrence and Victoria Justice, after breaching their iCloud accounts. A report on The Next Web subsequently linked the incident to a malicious script, which was reportedly uploaded to the website GitHub last month.

Same Origin Policy

- What is SOP?
  - Web Application Security Model
  - Policy enforced by browser
  - Constrained to **origin**: protocol, port, hostname

URL	Outcome	Reason
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	Different protocol
http://store.company.com:81/dir/etc.html	Failure	Different port
http://news.company.com/dir/other.html	Failure	Different host

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

SOP Protection

- Protects foreign requests from executing in your authenticated session as long as the foreign request is coming from a different **origin**.

- Example:
  - 1) User logged into https://mybank.com
  - 2) Opens tab to vulnerable site which has planted XSS; The XSS injects malicious iFrame into user's session in other tab: https://mybank.com
  - 3) SOP stops this attempt (different hostname, different protocol)

SOP Caveat

- SOP is great however, it will NOT protect you against externally referenced images, styles and scripts!
- External scripts are allowed by SOP!
- Why? SOP sees does not view these components (js, img, css) as "data" so allows access to foreign sites and their execution

Bypassing SOP

- Implement any operation (e.g., Click buttons) on the user's behalf
  - Using **JSONp**, see BlackHat Europe 2014 Talk by Ben Hayak
  - CallBacks
    - Legitimately used by Google and others to share data
    - Can become the injection points for an attacker
    - Any page on the domain becomes vulnerable

Defenses & Countermeasures

- Content Security Policy
- Secure HTTP Headers
- HTML5 Whitelisting

What is Content Security Policy?

- Content Security Policy (CSP) is a whitelist you can define in your web application to authorize the execution of scripts
  - Delivered via HTTP Header (configure web server or programmatically add)
  - Allows whitelisting of approved sources of content that browser may load including JavaScript and Cascading Stylesheets
  - Its like a cheap/poor man's version of a Web Application Firewall (WAF) for injection-related attacks

Why should I care about Content Security Policy?

- Effective countermeasure to XSS attacks, which usually lead to CSRF attacks
- Protects the DOM, prevents data leakage, protects against AJAX attacks
- Protects against externally referenced images, styles and scripts which Same Origin Policy (SOP) does not do
- Protects against iFrame injection (i.e., clickjacking)

Can I see an example of CSP?

- **Example:**

```
Content-Security-Policy: script-src 'self'
```

- This CSP specifics that **only content from this website** is allowed to execute, including externally referenced images, styles and scripts

Are there cost-efficiencies to be gained by using CSP?

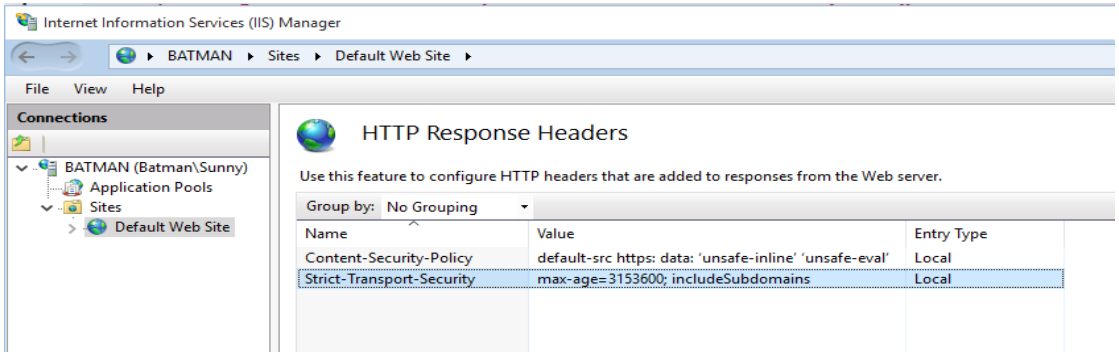
- YES!
- CSP protects your entire web application and all subdomains (so long as you specify).
  - This means it will protect areas of your web application inadvertently missed by programmers in their whitelisting techniques.
  - It will protect areas of your web application where vulnerabilities may reside that are not detected by your static code analyzer (e.g., HP Fortify).
  - It will protect areas of your web application inadvertently not tested by web app pen-testers

- CSP provides mitigation techniques that can save money in the following areas:
  - Pen-test remediation costs, including QA and Deployment costs
  - Static code analyzer mitigation development costs related to injection-type attacks (SQLi, iFrame, clickjacking, XSS, etc.)

How do I implement CSP?

Several Options Available including the following:

1. IIS Configuration



2. Apache Configuration

Apache: `Header always set Content-Security-Policy "default-src https: data: 'unsafe-inline' 'unsafe-eval'"`

3. Programmatically
   - Any programming language providing the ability to set HTTP Response headers can be used
   - Example shown is Java:

   - Full Java Servlet example here: https://www.owasp.org/index.php/Content_Security_Policy

```
// Define list of CSP HTTP Headers
this.cspHeaders.add("Content-Security-Policy");
this.cspHeaders.add("X-Content-Security-Policy");
this.cspHeaders.add("X-WebKit-CSP");
```

What directives are available in CSP?

```
default-src: Define loading policy for all resources type in case of a resource type dedicated directive is no
t defined (fallback),
script-src: Define which scripts the protected resource can execute,
object-src: Define from where the protected resource can load plugins,
style-src: Define which styles (CSS) the user applies to the protected resource,
img-src: Define from where the protected resource can load images,
media-src: Define from where the protected resource can load video and audio,
frame-src: Define from where the protected resource can embed frames,
font-src: Define from where the protected resource can load fonts,
connect-src: Define which URIs the protected resource can load using script interfaces,
form-action: Define which URIs can be used as the action of HTML form elements,
sandbox: Specifies an HTML sandbox policy that the user agent applies to the protected resource,
script-nonce: Define script execution by requiring the presence of the specified nonce on script elements,
plugin-types: Define the set of plugins that can be invoked by the protected resource by limiting the types of
resources that can be embedded,
reflected-xss: Instructs a user agent to activate or deactivate any heuristics used to filter or block reflect
ed cross-site scripting attacks, equivalent to the effects of the non-standard X-XSS-Protection header,
report-uri: Specifies a URI to which the user agent sends reports about policy violation
```

If I implement CSP, will my web page code break?

- Any inline JS or inline CSS calls would be broken unless you use `unsafe-inline` directive but I recommend against using the directive since it will allow attacker-controlled scripts to execute on your website. You can use a nonce or hashed-values for inline JS or CSS exceptions, if you like.

- Any existing inline JS or inline CSS needs to be externalized to a JS or CSS file and referenced in your web page by using the explicit <script> tags.

- For example, if you have a block of JS code for Google Analytics, you would have to create an external file and reference it like this:
  - **<script src="/assets/js/ga.min.js"></script>**

- Also, any inline event handlers like onClick"doMyStuff();" have to be removed and replaced with addEventListener() calls instead.

What does CSP look like from a client browser perspective?

Presentation Pointer LiveZoom

35 requests | 147 KB transferred | Finish: 1.72 s | DOMContentLoaded: 869 ms | Load: 1.33 s

Console Search Emulation Rendering

<top frame> Preserve log

```
Refused to load the script 'http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.5.2.min.js' becaus
Resource interpreted as Stylesheet but transferred with MIME type text/plain: "http://blogs.m
Refused to execute inline script because it violates the following Content Security Policy di
DuxsenS4SxbGcSjySJ1BA='), or a nonce ('nonce-...') is required to enable inline execution.
Refused to load the script 'http://i1.blogs.msdn.com/rrcontent/e3f41b57b5f074ccd4e6712a8353b9
Policy directive: "script-src 'none'".
Refused to execute inline script because it violates the following Content Security Policy di
6ByjMYvjOmo3q5aq3yxv-X-9LN6HyD8470='), or a nonce ('nonce-...') is required to enable inline
Refused to execute inline script because it violates the following Content Security Policy di
6hUxt4ISbfejzYOPmrOERIjeIqDhPoaIYKU0Unf8ew8=), or a nonce ('nonce-...') is required to enabl
Refused to load the script 'http://i4.services.social.microsoft.com/Search/Widgets/SearchBox.
following Content Security Policy directive: "script-src 'none'".
Refused to load the script 'http://i4.services.social.microsoft.com/Search/Widgets/SearchBox.
following Content Security Policy directive: "script-src 'none'".
Refused to load the script 'http://i1.blogs.msdn.com/rrcontent/47499858c4077d6e043f7add140009
Policy directive: "script-src 'none'".
Refused to load the script 'http://i1.blogs.msdn.com/rrcontent/c6cf9e2cff3f0bfc0a1a8c20bf2435
Policy directive: "script-src 'none'".
Refused to execute inline script because it violates the following Content Security Policy di
VSlNlHzpC6AWzS1EZsBHPiYV2JR10yCg='), or a nonce ('nonce-...') is required to enable inline ex
Refused to load the script 'http://i1.blogs.msdn.com/rrcontent/2569210a296af7a76a1f0a41cafcfc
```

Which browsers are compatible with CSP headers?

- Full compatibility table here: http://caniuse.com/contentsecuritypolicy

IE	Edge			Safari		iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android
		29	34		20									
		30	35	3.1	21			2.1						
		31	36	3.2	22	iOS Safari		2.2						
IE		32	37	4	23	3.2		2.3		Opera Mobile				
5.5		33	38	5	24	4.0-4.1		3		10				
6		34	39	5.1	25	4.2-4.3		4		11				
7		35	40	6	26	5.0-5.1		4.1		11.1				
8		36	41	6.1	27	6.0-6.1		4.2-4.3		11.5				
9		37	42	7	28	7.0-7.1		4.4	Blackberry Browser	12			IE Mobile	UC Browser for Android
10	Edge	38	43	7.1	29	8	Opera Mini	4.4.3-4.4.4	7	12.1	Chrome for Android	Firefox for Android	10	
11	1	39	44	8	30	8.1-8.4	5.0-8.0	40	10	30	42	39	11	9.9
		40	45	9	31	9								
		41	46		32									
		42	47											

Can I watch a demo to see how CSP works?

Yes!

Are there other HTTP Response Headers available that can protect my web application?

- Yes!
- In addition to Content-Security-Policy, you may add these additional security-related HTTP Response Headers:
  - ***HTTP Strict Transport Security***
    - To ensure that users of your site must always use HTTPS, add this header. It will even work on old bookmarks, forcing users to instead use HTTPS.
  - ***HTTP Public Key Pinning***
    - To ensure that only YOUR server's TLS digital certificate is authorized for client browsers to trust, add this header. This prevents attacker-controlled certificates for your server (should the CA be compromised) from being accepted by clients.
  - ***X-Frame Options***
    - To ensure that no malicious iFrames are loaded or executed on your website; protects against clickjacking attack.
  - ***X-XSS Protection***
    - Ensures the use of built-in browser protection against XSS attacks. Settings are 0 (disable) and 1 (enable) with a `1; mode=block` telling the browser to block the execution of a script if it detects an attack.
  - ***X-Content-Type Options***
    - Provides the `nosniff` directive the sniffing of the mime-type for an uploaded file. By not allowing this sniff to occur, this mitigates spoofing of the content-type to circumvent whitelisting techniques within the application code.

X-FRAME Header Options

- SAMEORIGIN
- DENY (Recommended)
- ALLOW-FROM: <explicit domain>
- https://www.owasp.org/index.php/List_of_useful_HTTP_headers
- Protects against Clickjacking (injection of iFrames)

HTML 5 Whitelisting

- Never allow client-side callback functions
- Whitelist callback domains, redirects always on server-side

References

- BlackHat 2014 Talk: Same Origin Method Execution (Ben Hayak): https://www.youtube.com/watch?v=UfYfID_r7-U

- Defcon 21 Talk: How to use CSP to stop XSS (Ken Lee): https://www.youtube.com/watch?v=BEsEIV8v2fQ