

Using HTML5 To Make JavaScript (Mostly) Secure

Mike Shema
Hacker Halted US
September 20, 2013



Hello Again, Atlanta!



A Definition

Ja·va·Script | 'jävəskript |

invective.

1 A vendor-neutral*, cross-platform liability for generating asynchronous, event-driven browser bugs.

2 Interpreted language for exploiting string concatenation of HTML.

* mostly

Subtle and Quick to Danger

- Programming traps
 - Scope, blocks, & var
 - Types & type coercion
 - Manipulating the DOM
- Expanding from client to server
 - Echoes of PHP

Subtlety Gradient

```
document.write(document.location.href)
```

```
typeof null == "object";  
typeof undefined == "undefined"  
null == undefined;  
null == undefined; // nope!
```

```
(window[(![]+[])[1] + (![]+[])[2] + (![]+[])[4] +  
  (!![]+[])[1] + (!![]+[])[0]  
  ])(9)
```

JavaScript Crypto



- Use TLS for channel security
 - Better yet, use HSTS and DNSSEC.
- No trusted execution environment in...
 - ...the current prototype-style language
 - ...an intercepted HTTP connection
 - ...an exploitable HTML injection vuln

JavaScript Crypto



- `Math.random()`

- `sjcl.random`

- Fortuna-like generator



JavaScript Crypto



- **Minimize lifetime of plaintext password**
 - Client-side PBKDF2
 - Challenge-response
- **...but possibly lose some security insights**
 - Password composition, history
 - Patterns of brute force activity



`<!doctype html>`

Browser Security Confidence

- Countermeasure

- Process separation

- Sandboxing plugins

- XSS Auditors

- Phishing warnings

- Auto-updating

- Half-Life

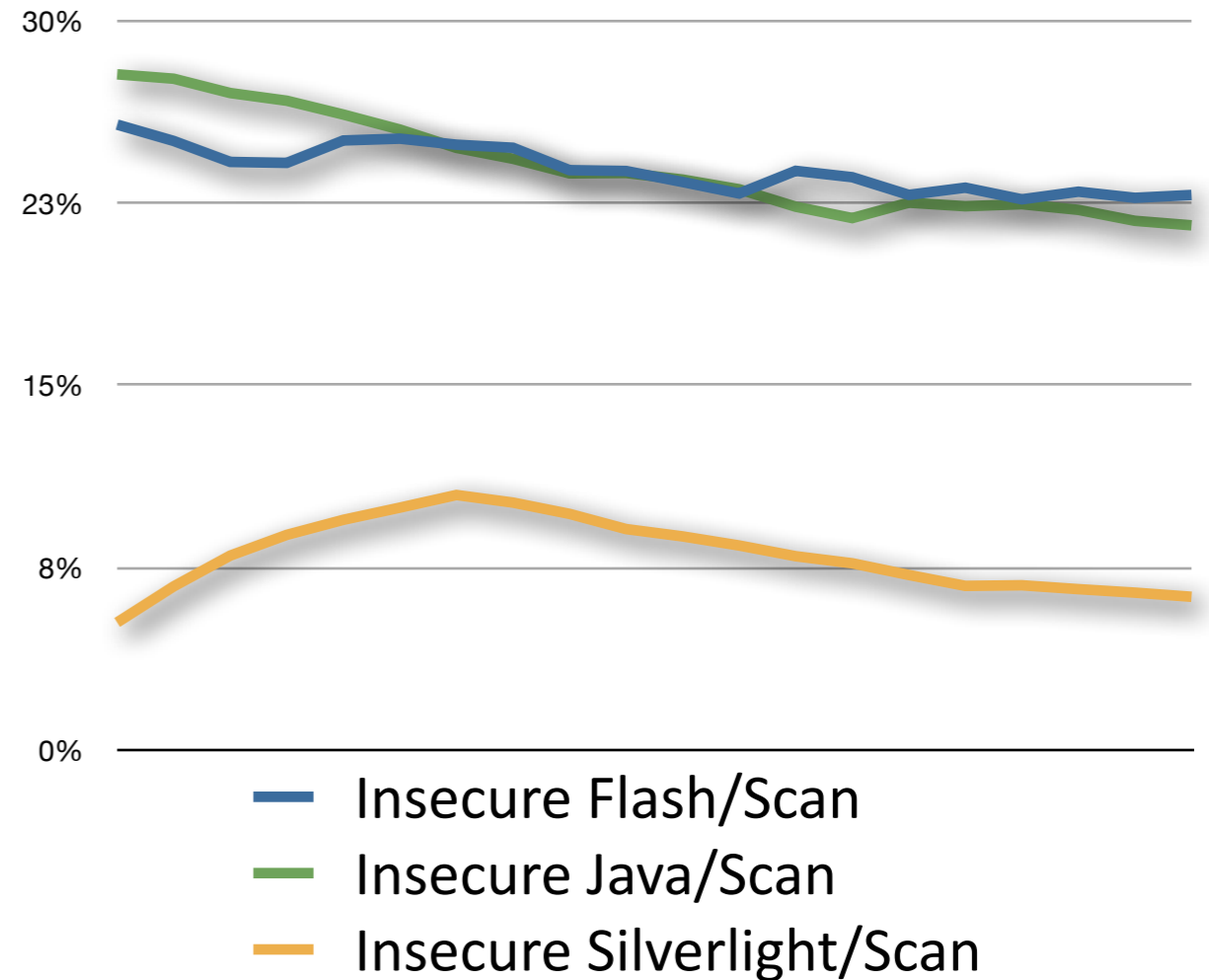
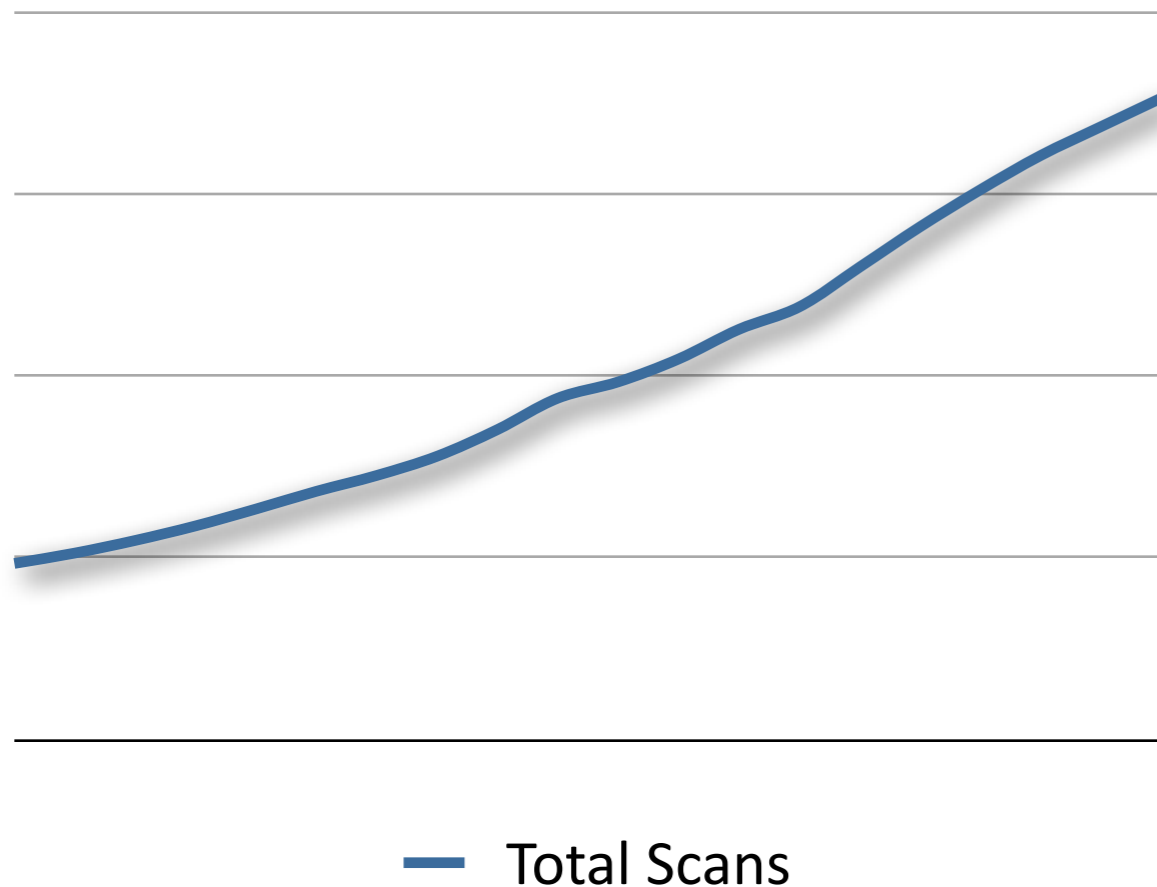
- 1 year

- 6 months

- 3 months

- 6 weeks

Software Expiration



“Emphasize freshness date over version number.”

HTML Injection


- The 20+ year-old vuln that refuses to die.
- But JavaScript makes the situation better!
- No, JavaScript makes the situation worse!
- HTML5 to the rescue!?



Oh, No! XSS Is Worse!

```
http://web.site/vuln?foo=xss"...
```

```
<input type="text" name="foo"  
value="xss" autofocus  
onfocus=alert(9);//">
```



(yawn)

XSS Blacklisting Is Worse

- New elements, new attributes require new patterns
- Security through Regexity tends to fail...

```
<img src="" onerror=alert(9)>  
  
<a href="" &<img& /onclick=alert(9)>foo</a>  
<script/<a>alert(9)</script>  
<script/<a>alert(9)</script <a>foo</a>  
<script%20<!--%20->alert(9)</script>
```

Client-Side Validation

4.10.7 The input element — HTML Standard

Keyword	State	Data type
hidden	Hidden	An arbitrary string
text	Text	Text with no line breaks
search	Search	Text with no line breaks
tel	Telephone	Text with no line breaks
url	URL	An absolute URL
email	E-mail	An e-mail address or list of e-mail addresses
password	Password	Text with no line breaks (sensitive information)
datetime	Date and Time	A date and time (year, month, day, hour, minute, second, fraction of a second) with the time zone as UTC
date	Date	A date (year, month, day) with no time zone
month	Month	A date consisting of a year and a month with no time zone
week	Week	A date consisting of a week-year number and a week number with no time zone
time	Time	A time (hour, minute, seconds, fractional seconds) with no time zone

4.10.7 The input element — HTML Standard

datetime-local	Local Date and Time	A date and time (year, month, day, hour, minute, second, fraction of a second) with no time zone
number	Number	A numerical value
range	Range	A numerical value, with the extra semantic that the exact value is not important
color	Color	An sRGB color with 8-bit red, green, and blue components
checkbox	Checkbox	A set of zero or more values from a predefined list
radio	Radio Button	An enumerated value
file	File Upload	Zero or more files each with a MIME type and optionally a file name
submit	Submit Button	An enumerated value, with the extra semantic that it must be the last value selected and initiates form submission
image	Image Button	A coordinate, relative to a particular image, with the extra semantic that it must be the last value selected and initiates form submission

Sophisticated Exploits

The screenshot displays the BeEF Control Panel interface. The window title is "BeEF Control Panel". The top navigation bar includes "BeEF 0.4.4.8-alpha", "Submit Bug", and "Logout". The left sidebar shows a tree view of "Hooked Browsers" with categories "Online Browsers", "Offline Browsers", and "localhost". Under "localhost", several browser instances are listed, all with the IP address "127.0.0.1". The main content area is divided into tabs: "Getting Started", "Logs", and "Current Browser". The "Current Browser" tab is active, showing a "Details" sub-tab. The details are organized into three categories:

- Category: Browser (5 Items)**
 - Browser Name: Opera Initialization
 - Browser Version: 12 Initialization
 - Browser UA String: Opera/9.80 (Macintosh; Intel Mac OS X 10.8.2; U; en) Presto/2.10.289 Version/12.02 Initialization
 - Browser Plugins: navigator.plugins is not supported in this browser! Initialization
 - Window Size: Width: 300, Height: 150 Initialization
- Category: Browser Components (9 Items)**
 - Flash: Yes Initialization
 - Java: Yes Initialization
 - VBScript: No Initialization
 - PhoneGap: No Initialization
 - Google Gears: No Initialization
 - Web Sockets: Yes Initialization
 - ActiveX: No Initialization
 - Session Cookies: Yes Initialization
 - Persistent Cookies: Yes Initialization
- Category: Hooked Page (5 Items)**
 - Page Title: No Title Initialization
 - Page URI: http://localhost/ch2/BeEF/infected.html Initialization
 - Page Referrer: http://localhost/ch2/BeEF/csp_no_iframe.php Initialization

At the bottom left, there are tabs for "Basic" and "Requester".

Same Vulns, New Exploits

```

```

```
<link rel="prefetch" href="https://
csrf.target/sensitive?action=something">
```

- ~~Origin~~
- Referer
- X-Moz: prefetch

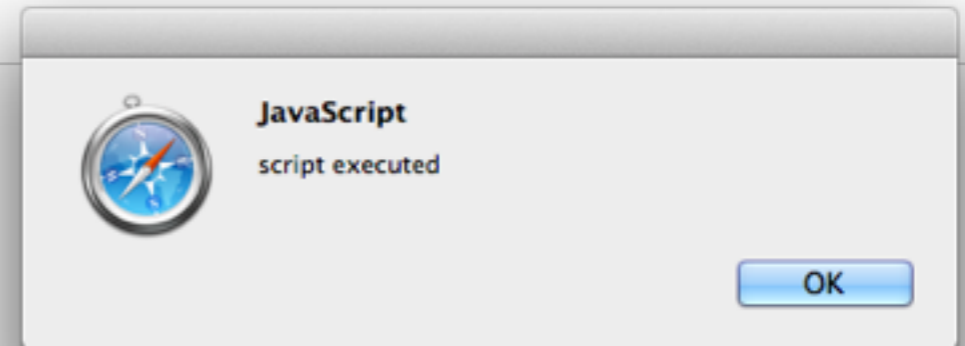
Improving SOP

- Granular access control
 - Whatever happened to least privilege?
- Make the `<iframe>` more useful for isolating Origins
 - seamless
 - sandbox



`<iframe * src="infected.html">`

`(empty)`



`sandbox`

JavaScript not executed

`sandbox="allow-scripts"`

JavaScript executed
~~document.cookie~~
~~localStorage()~~
~~sessionStorage()~~

`text/html-sandboxed`

Waiting for browser support

On the Other Hand...

...if you're relying on JavaScript frame-busting instead of X-Frame-Options: DENY.

```
function killFrames(){if(top.location!=location)
{if(document.referrer){var
a=get_hostname_from_url(document.referrer);var
b=a.length;if(b==8&&a!="web.site")
{top.location.replace(document.location.href)}else
if(b!=8&&a.substring(a.length-9)!=".web.site")
{top.location.replace(document.location.href)}}}
if(top.frames.length!
=0)top.location=self.document.location}function
get_hostname_from_url(a){return a.match(/:\//(. [^/?]
+)/)[1]}killFrames();
```

Content Security Policy

- Granular access for retrieving resources
- Declared by header directives
 - Will require code changes, or unsafe-inline
- Waiting for universal implementation
 - And new versions being defined
- <http://www.w3.org/TR/CSP/>



Selective Resource Control

```
X-CSP: default-src 'self'; frame-src 'none'
```

```
<!doctype html>  
<html>  
<body>  
    <iframe src="/infected.html"></iframe>  
</body>  
</html>
```

Defeat Exploits, Not Vulns

```
X-CSP: default-src 'self'
```

```
<input type="text" name="q" value="foo"  
autofocus onfocus=alert(9)//>
```

```
X-CSP: default-src 'self' 'unsafe-inline'
```

```
<input type="text" name="q" value="foo"  
autofocus onfocus=alert(9)//>
```

[https://web.site/page#<img/src=""onerror=alert\(9\)>](https://web.site/page#<img/src=)

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-1.8.2.min.js"></script>
<script>
$(document).ready(function() {
  var x = (window.location.hash.match(/^#([^\./].+)$/)) || [];
  var w = $('a[name="" + x + "'], [id="" + x + "']');
});
</script>
</head>
<body>
  <div id="main">foo</div>
</body>
</html>
```


[https://web.site/page#<img/src=""onerror=alert\(9\)>](https://web.site/page#<img/src=)

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-1.8.2.min.js"></script>
<script src="main.js"></script>
</head>
<body>
  <div id="main">foo</div>
</body>
</html>
```

```
$(document).ready(function() {
  var x = (window.location.hash.match(/^#([^\./].+)$/)) || []][1];
  var w = $('a[name="" + x + "]", [id="" + x + "]);
});
```

Decouple HTML & JS

- Avoid “inline” event handler attributes

```
$('#main').attr('onclick',  
'alert(9)');
```

- Use event managers

```
$('#main').bind("click",  
function(e) { alert(9) });
```

```
$('#main').click(function(e)  
{ alert(9) });
```

```
$('#main').on("click",  
function(e) { alert(9) });
```

On the Other Hand...

...an awesome XSS DoS payload if injectable into a `<head>` section.

```
<meta http-equiv="X-WebKit-CSP"
content="default-src 'none'">
```

On the Other Hand...

...another way to forge POST method for CSRF.

```
<!doctype html><html><head>  
<meta http-equiv="X-WebKit-CSP"  
      content="img-src 'none'; report-uri  
'https://csrf.target/page?a=1&b=2&c=3'">  
</head><body>  
  
</body></html>
```

Partial CSRF Influence

```
POST /page?a=1&b=2&c=3 HTTP/1.1
Host: csrf.target
User-Agent: Mozilla/5.0 ...
Content-Length: 116
Accept: */*
Origin: null
Content-Type: application/x-www-form-urlencoded
Referer: http://web.site/HWA/ch3/csrf.html
Cookie: sessid=12345
Connection: keep-alive
```

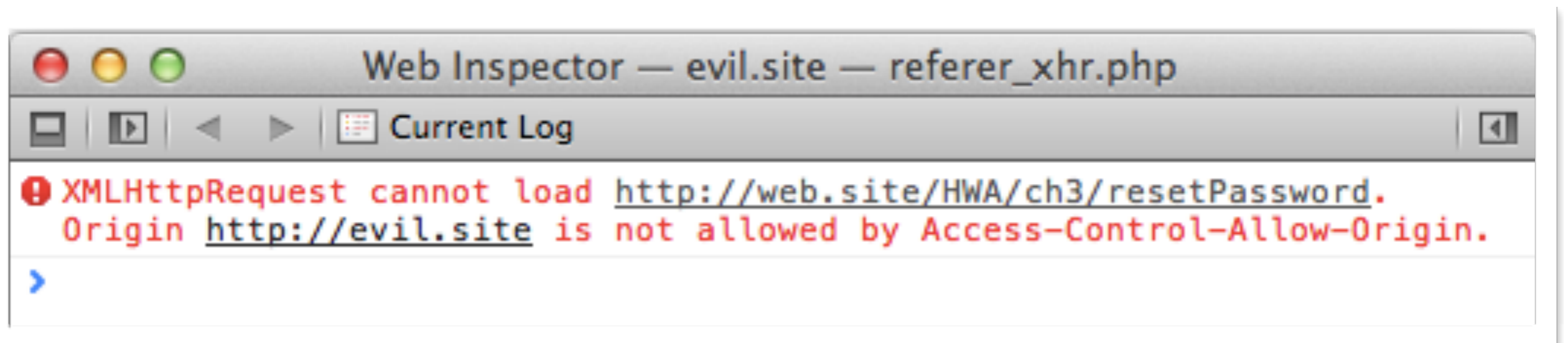
```
document-url=http%3A%2F%2Fcsrf.target%2FHWA%2Fch3%2Fcsrf.html&violated-directive=default-src+%27none%27
```

CORS

- Defines read-access trust of another Origin
 - Expresses trust, not security
 - But still contributes to secure design
- Principle of Least Privilege
 - Beware of Access-Control-Allow-Origin: *
 - Short Access-Control-Max-Age
 - Minimal Access-Control-Allow-{Methods | Headers}
- Verify the Origin

CORS Can Counter CSRF

- Create “non-simple” XHR requests
 - X-CSRF header
 - Inhibit forgery (creation)



CORS Can Counter CSRF

- Refactor content to broker requests through XHR.
 - No nonces, no tokens
 - ...but doesn't work for legitimate non-origin incoming requests
 - ...and requires HTML5 browsers

WebSockets



- New ~~protocol!~~ *fuzzing target*
- Excellent covert channel
 - Masking, compression complicates inspection
 - Data frames can be sneaky
- Solves connection, not security, problems

Capability, Security, Privacy*

“In a world with one eye on privacy, the blind browser is king.”

- AppCache
- Battery Status
- Geolocation
- Web Storage
- WebGL
- WebPerf APIs
- Browser Fingerprinting
- Device Fingerprinting
- Usage Statistics
- User Tracking

* choose two (one?)

Privacy

- **Implementation vs. design**
 - Specs that acknowledge areas of concern
- **Browser Fingerprinting**
- **Inference-based attacks**
 - Timing, cache
- **Data exposure**
 - Web Storage API

“And what does it say now?” asked Arthur.

“*Mostly harmless,*” admitted Ford with a slightly embarrassed cough.

end.isNigh()

JavaScript Will Improve

- Libraries driving good design patterns
 - ...and moving to be compatible with CSP
- Steps towards a trusted environment
 - Freeze & Seal an Object
 - `Object.hasOwnProperty()`
 - Modular libraries
 - `toStaticHtml()`✖

Careful Implementation

- Origin is an identity hint, not an access control attribute
 - The return of *X-Forwarded-For*
- JSON serializes, not sanitizes, data
- Avoid string concatenation
 - Review, refactor, refine

Rely on Security from Design

- Strong solutions
 - SQL injection -- prepared statements
 - Clickjacking -- X-Frame-Options
- Mitigating solutions
 - HTML injection -- Content Security Policy
 - Mixed-Origin content -- CORS, CSP, <iframe> sandbox
 - Sniffing -- HSTS
- Implementation-specific solutions
 - CSRF -- hmm...*

* <https://github.com/mutantzombie/SessionOriginSecurity>

Trends to Discourage

- “Legacy” support of draft protocol versions
 - WebSockets, CSP iterations
- Storing personal data in the browser
 - One XSS away (or malware, or...)
- Ever-changing specs...
 - At least, those that lead us back to quirks
- More plugins

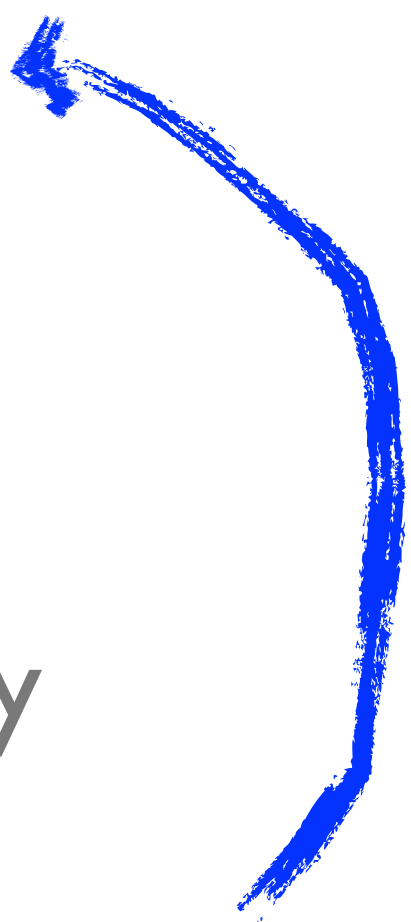
Trends to Encourage

- Compartmentalized plugins
 - Per domain, per origin
- Enable SOP to be more granular
- Enable mixed-origin content to be more secure
- Security from design
 - Better than ad-hoc implementation

Steps to Take

- Use HTTPS everywhere
 - Prep for HSTS
- Decouple HTML & JavaScript
 - Prep for CSP without unsafe-inline
- Sandbox content
 - Use even more iframes

Code Like It's Not 1999

- Encourage users to update browsers
 - Disable plugins, become secure
 - Design web apps for data security
 - Design web browsers for data privacy
 - Adopt HTML5 security features
 - ...to protect users with HTML5-enabled browsers
- 

Thank You!

Questions?

- @CodexWebSecurum
- <http://deadliestwebattacks.com>
- *Hacking Web Apps*



“Gutenberg Injection”

Book

cessfully render following `` element:

```
<img/src="." alt="" onerror="alert('zombie')"/>
```

JavaScript doesn't have to rely on quotes to establish strings

JSON

```
{..., "totalResults": 4, "results":  
[[...], [...], [33, "Page 16", "... t  
require spaces to delimit their  
attributes. <img/src=\".\" alt=  
\"\" onerror=\"alert('<b>zombie</  
b>')\"/\"/> JavaScript doesn't have  
to rely on quotes to establish  
strings, nor do ...\", ...]]}
```

DOM

```
...>Page 16</span> ... t  
require spaces to delimit their  
attributes.  JavaScript  
doesn't have to...
```

Here, There, Everywhere

- **asm.js** [<http://asmjs.org>]
- **jQuery** [<http://jquery.com>]
- **pdf.js** [<http://mozilla.github.com/pdf.js/>]
- **sjcl.js** [<http://crypto.stanford.edu/sjcl/>]
- **BeEF** [<http://beefproject.com>]
- **Screen Shots** [<https://github.com/niklasvh/html2canvas>]