Using HTML5 To Make JavaScript (Mostly) Secure

Mike Shema
Hacker Halted US
September 20, 2013
Hello Again, Atlanta!
A Definition

JavaScript | jävəskrɪpt |

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 Anger

• 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!

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.”

20 months of QBC data starting November 2011
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...

```html
<img src="" onerror=alert(9)>
<img src=""a="" onerror=alert(9)>
<a href=""&<img amp;/onclick=alert(9)>foo</a>
<script><a>alert(9)</a></script>
<script><a>alert(9)</a></script> <a>foo</a>
<script%20<!–%20–>alert(9)</script>
```
### Client-Side Validation

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

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

```html
<img src="https://csrf.target/sensitive?action=something">

<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
<table>
<thead>
<tr>
<th>sandbox attribute</th>
<th>JavaScript behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>(empty)</td>
<td>JavaScript not executed</td>
</tr>
<tr>
<td>sandbox</td>
<td>JavaScript executed</td>
</tr>
<tr>
<td>sandbox=&quot;allow-scripts&quot;</td>
<td>JavaScript executed, but certain features like <code>document.cookie</code>, <code>localStorage()</code>, and <code>sessionStorage()</code> are disabled</td>
</tr>
<tr>
<td>text/html-sandboxed</td>
<td>Waiting for browser support</td>
</tr>
</tbody>
</table>
On the Other Hand...

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

```javascript
function killFrames()
{
if (top.location != location)
{
if (document.referrer)
{
var a = getHostnameFromUrl(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
}
return a.match(/:\//\//\(\.[^/\?] +\)/)[1]
}
```
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'

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

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

```html
<input type="text" name="q" value="foo" autofocus onfocus=alert(9)//"/>"
```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-1.8.2.min.js"></script>
<script>
$(document).ready(function() {
    var x = (window.location.hash.match(/^[^#][^\/]\.(.+)$/) || []).length;
    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)">

```html
<!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>

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

• Avoid “inline” event handler attributes

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

• Use event managers

```javascript
$('#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.

```html
<meta http-equiv="X-WebKit-CSP" content="default-src 'none'" />
```
On the Other Hand...

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

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

- 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.
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”

JavaScript doesn’t have to rely on quotes to establish strings, nor do...

... require spaces to delimit their attributes. 

```json
{...
"totalResults":4,"results":
[[...],[...],[33,"Page 16","... 
... require spaces to delimit their attributes. <img src="." alt="" 
onerror="alert('&lt;b&gt;zombie&lt;/b&gt;')"> JavaScript 
... doesn't have to...```
Here, There, Everywhere

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