BLUEPRINT: Robust Prevention of Cross-site Scripting Attacks for Existing Browsers

Mike Ter Louw       V.N. Venkatakrishnan

Presented by: Victor Parece
Cross-site Scripting (XSS)

- Injection of untrusted code into web page by user
- Injected script is run by victim's web browser within the context of the trusted web application
  - Persistent / Non-persistent
Vulnerable Web Applications

- Potentially vulnerable applications allow user to submit HTML content and then this content is output to the user's web browser.
- User's are visiting trusted website, but this site could contain untrusted code.
- Potential to attack large number of users.
Defense Approaches

• Content Filtering
  – Web application attempts to remove all scripts from user submitted content, while allowing benign HTML

• Browser Collaboration
  – Web application collaborates with web browser. User submitted content is marked as untrusted and browser does not execute scripts in these sections.
Content Filtering

- Filter must understand how untrusted content is interpreted by a variety of web browsers
  - Web Browsers parse code differently and may have quirks
  - Browser quirks are bad parsing behavior that does not follow language standards or are not defined by standards (malformed HTML)
- Very difficult to implement a correct and complete content filter
Browser Parsing Quirks

(a) Benign HTML blog comment

```html
<p>
Here is a page you might find
<b>very</b> interesting:
<a href="http://www.cpsr.org">
  Link</a>
</p>
<p style="text-align: right;">
Respectfully,
Alice</p>
</p>
```

(b) Malicious HTML blog comment

```html
<p>
Here is a page you might find
<b ""><script>doEvil(...)</script>"very</b> interesting:
<a href="&amp;#14; javasc&amp;#x0A;ript:doEvil(...);">Link</a>
</p>
<p style="nop:expres/*/xss*/sion(doEvil(...))">
Respectfully,
Eve</p>
</p>
```
Browser Collaboration

- BEEP (Browser-Enforced Embedded Policies)
  - Server-browser protocol to identify untrusted scripts
  - Modifies browser to understand protocol and enforce policy of denying untrusted scripts
- Requires a new protocol to be defined and implemented by numerous web browsers
- Effective long term solution, but not practical for current threats
Objectives of BLUEPRINT

- Robust XSS protection (including browser quirks)
- Allow benign HTML content submitted by users
- Compatible with existing web browsers
Web Application

Web Browser
BLUEPRINT's Approach

- Web application encodes areas of untrusted user content
  - Alphabet is comprised of syntactically inert characters
  - Encoded data is processed as plaintext by web browser
- Trusted JavaScript library decodes untrusted user content and writes it to the document using safe DOM APIs
  - Safe APIs do not generate JavaScript parse nodes
Web Application

Untrusted Content is encoded by the Web Application

Web Browser
Web Application

Untrusted Content is encoded by the Web Application

Web Browser
Model

- A model defines a region containing user submitted content that has been encoded
- When a model is loaded by the browser, the model interpreter decodes the model, builds the parse tree, and replaces the model with the content

```html
<code style="display:none;" id="__bpl1">
  =Enk/sCkh1cmUgaXMgYSBwYWdlIHlvdSBt... 
  =SkKICAgICI+dmVyeQ==C/k/QIGh1bHBmd... 
  =ECg==C/Enk/gCiAgUmVzcGVjdGZ1bGx5L...
</code>
<script id="__bpl1s">
  __bp__.cxCPCData("__bpl", "__bpl1s");
</script>
```
BLUEPRINT Integration

- Consists of server side component and JavaScript library.
- Untrusted content location must be identified and modified to support automatic model embedding.
- Different contexts are provided to restrict data

```php
<?php foreach ($comments as $comment): ?>
  <li>
    <?php echo($comment); ?>
  </li>
<?php endforeach; ?>
```
Available Contexts

- Contexts specify where a model is embedded to support untrusted user content

<table>
<thead>
<tr>
<th>Context</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CXTITLE</td>
<td>Document title</td>
<td>&lt;title&gt; Profile for user: untrusted &lt;/title&gt;</td>
</tr>
<tr>
<td>CXPCDATA</td>
<td>Parsed character data (PCDATA)</td>
<td>&lt;p&gt; untrusted &lt;i&gt;content&lt;/i&gt; &lt;/p&gt;</td>
</tr>
<tr>
<td>CJSNUMBER</td>
<td>JavaScript numeric literal</td>
<td>var x = 10;</td>
</tr>
<tr>
<td>CJSSTRING</td>
<td>JavaScript string literal</td>
<td>var x = &quot;untrusted&quot;;</td>
</tr>
<tr>
<td>CDATA</td>
<td>Character data (CDATA)</td>
<td>&lt;![CDATA[ untrusted ]]&gt;</td>
</tr>
<tr>
<td>CATTRIBVAL</td>
<td>Element attribute value</td>
<td>&lt;a href=&quot;.../untrusted.html&quot;&gt; ... &lt;/a&gt;</td>
</tr>
<tr>
<td>CATTRIB</td>
<td>Element attribute</td>
<td>&lt;td align=&quot;center&quot; nowrap&gt; ... &lt;/td&gt;</td>
</tr>
</tbody>
</table>
Evaluation

- Tested effectiveness and performance using eight popular browsers
- Integrated BLUEPRINT into MediaWiki and WordPress web applications
- Tested all XSS attacks provided by the OWASP XSS Cheat Sheet (total of 94)
## Results

<table>
<thead>
<tr>
<th>Type of attack</th>
<th># of variations</th>
<th># defended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-site scripting</td>
<td>94</td>
<td>94</td>
</tr>
<tr>
<td>Other (non-XSS)</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>Informational</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>113</strong></td>
<td><strong>94</strong></td>
</tr>
</tbody>
</table>

![Average memory overhead with Blueprint enabled (%)](chart.png)
Results

WordPress

MediaWiki
Conclusion

- BLUEPRINT is an effective solution for stopping XSS attacks
  - Prevented all 94 attacks tested
  - Performance hit is relatively small
- BLUEPRINT provides defenses without requiring browser modification
- Browser Collaboration approach is better long term solution, since overhead would be even less
Citation
