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
- WordPress, Facebook, LiveJournal, and MySpace

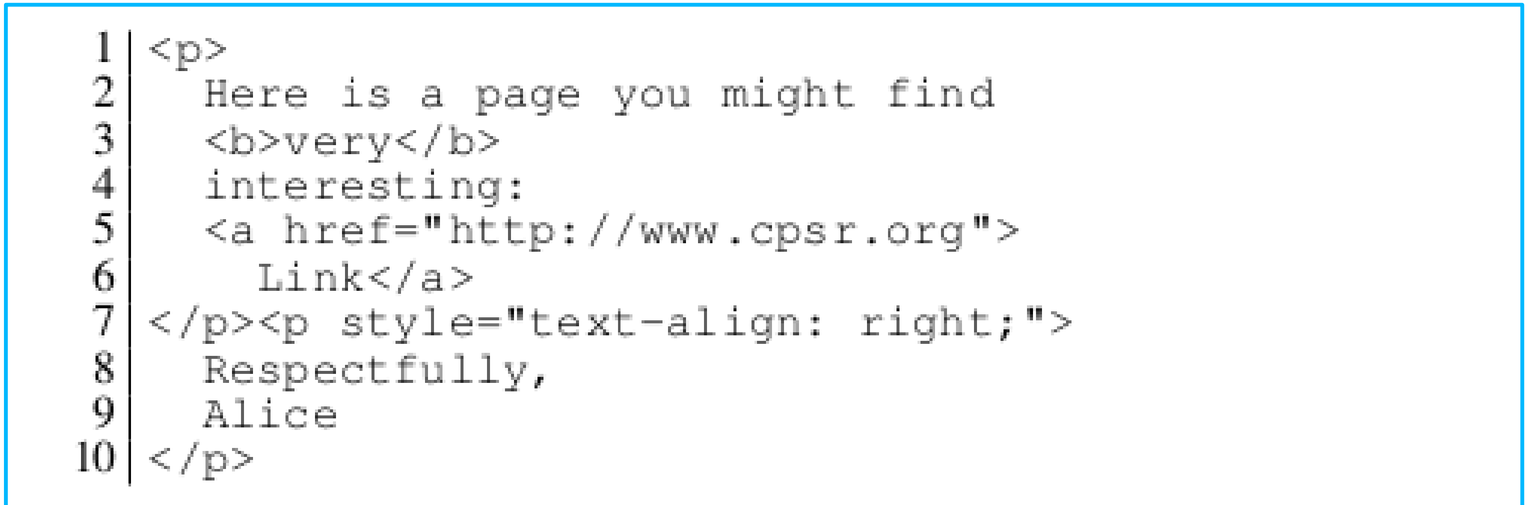
Defense Approaches

- 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

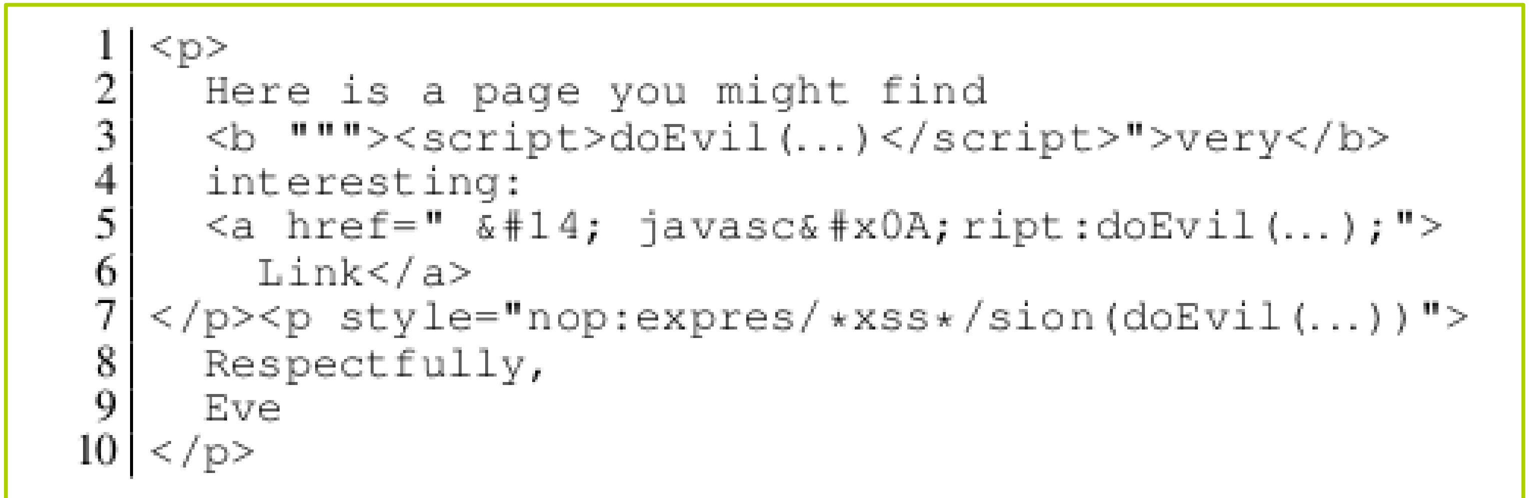
- Filter must understand how untrusted content is interpreted by a variety of web browsers
  - Web Browsers parse code differently and may have quirks
  - Browser quirks are bad parsing behavior that does not follow language standards or are not defined by standards (malformed HTML)
- Very difficult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

Browser Parsing Quirks



```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>
```

(a) Benign HTML blog comment



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

(b) Malicious HTML blog comment

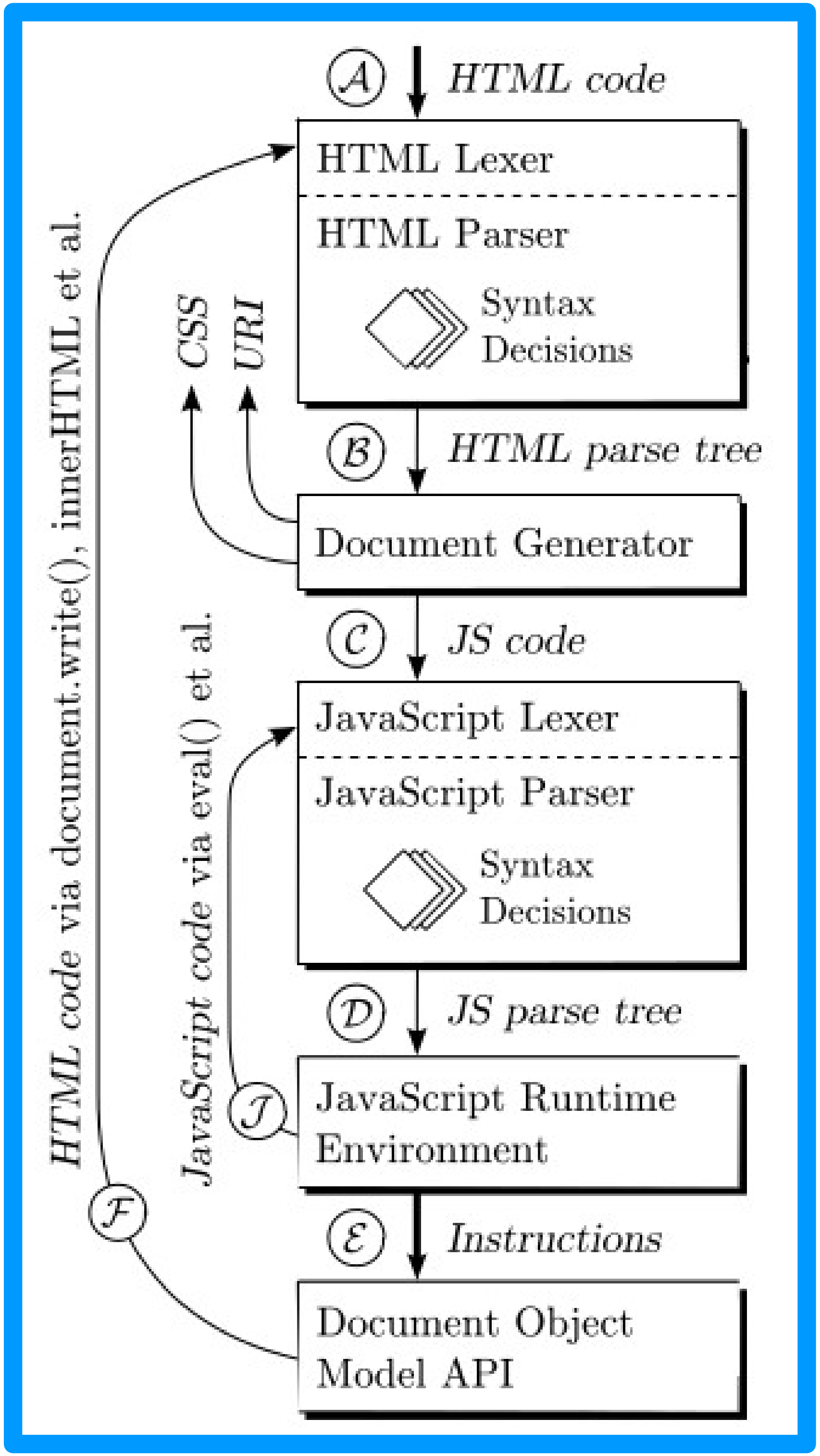
Browser Collaboration

- 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

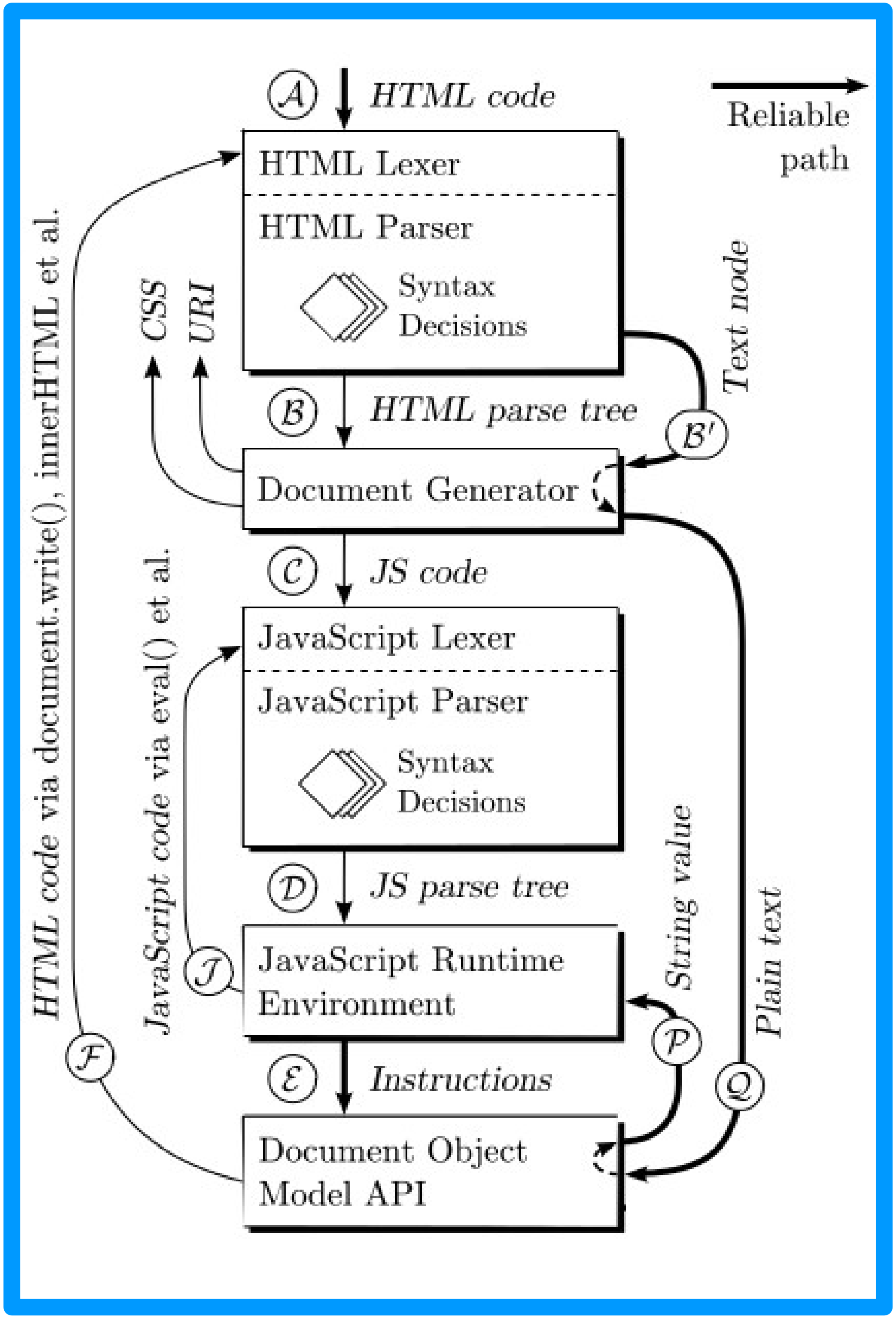
BLUEPRINT's Approach

- 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

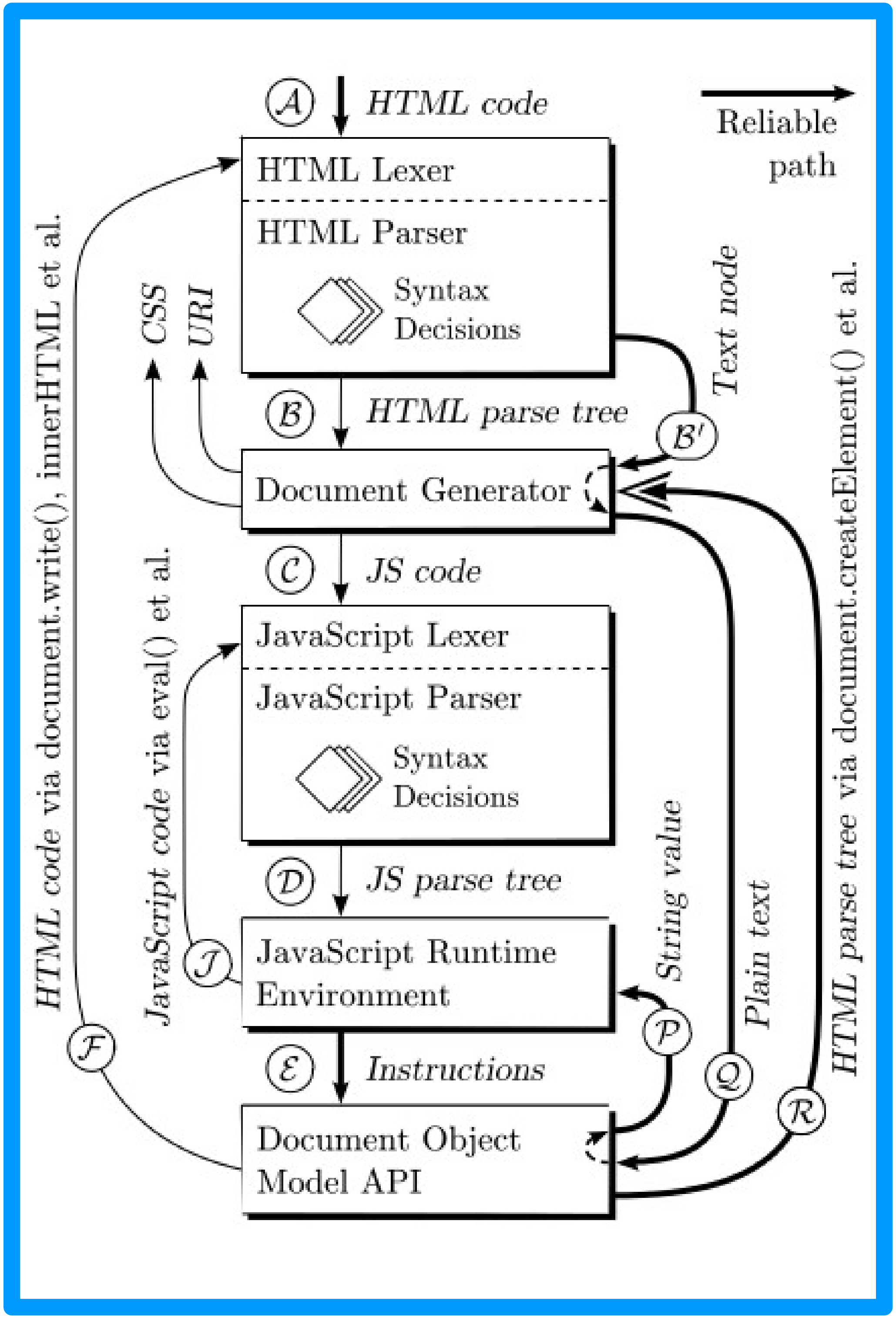
Untrusted Content
is encoded by the
Web Application

Web Browser



Web Application

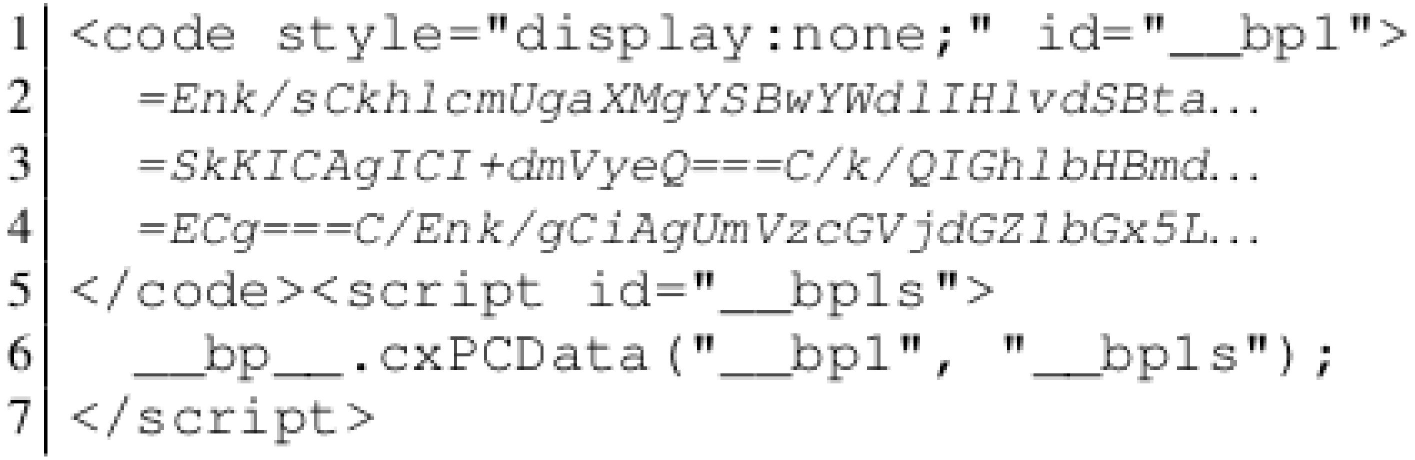
Untrusted Content
is encoded by the
Web Application



Web Browser

Model

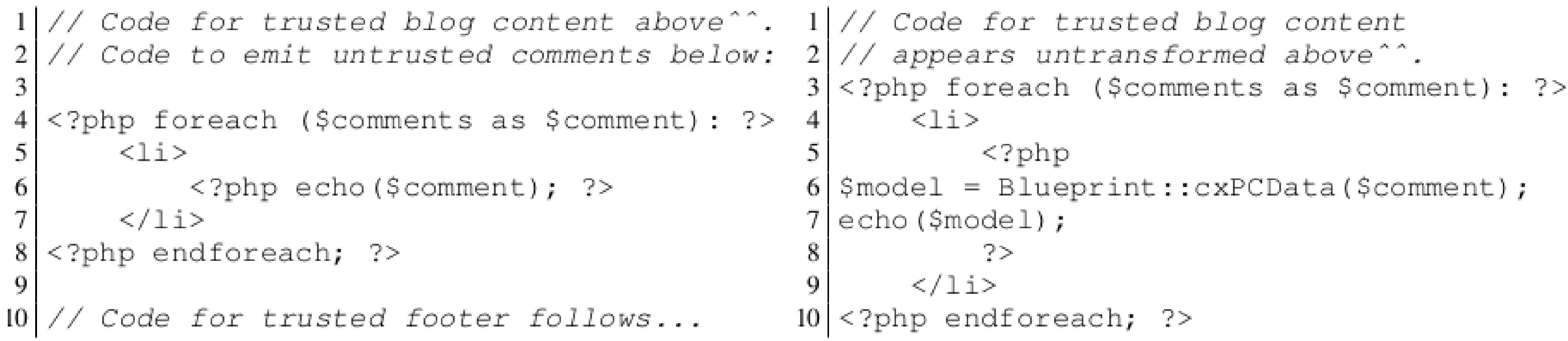
- A model defines a region containing user submitted content that has been encoded
- When a model is loaded by the browser, the model interpreter decodes the model, builds the parse tree, and replaces the model with the content



```
<code style="display:none;" id="__bp1">
   =Enk/sCkhlcmUgaXMgYSBwYWdlIHlvdSBta...
   =SkKICAgICI+dmVyeQ===C/k/QIGhlbHBmd...
   =ECg===C/Enk/gCiAgUmVzcGVjdGZlbGx5L...
</code><script id="__bp1s">
  __bp__.cxPCData("__bp1", "__bp1s");
</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

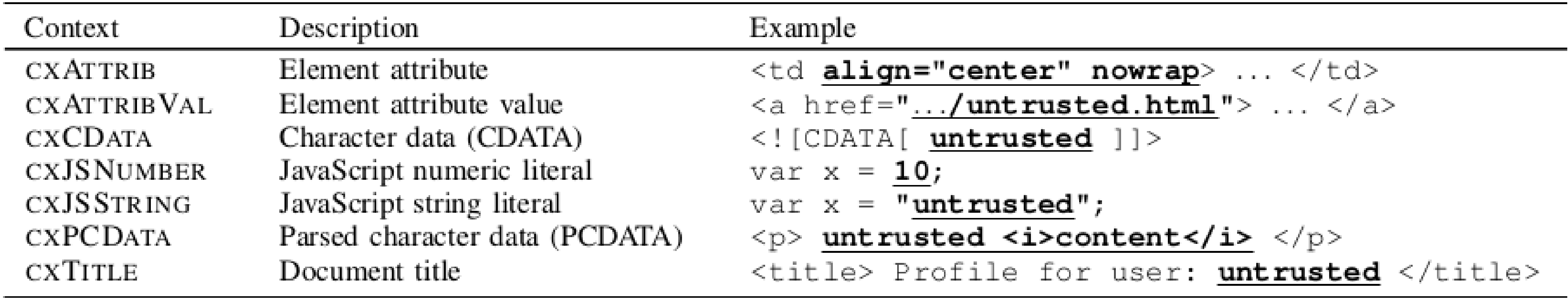


```
// Code for trusted blog content above^^.
// Code to emit untrusted comments below:

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

// Code for trusted footer follows...
```

```
// Code for trusted blog content
// appears untransformed above^^.
<?php foreach ($comments as $comment): ?>
    <li>
        <?php
$model = Blueprint::cxPCData($comment);
echo($model);
        ?>
    </li>
<?php endforeach; ?>
```

Available Contexts

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

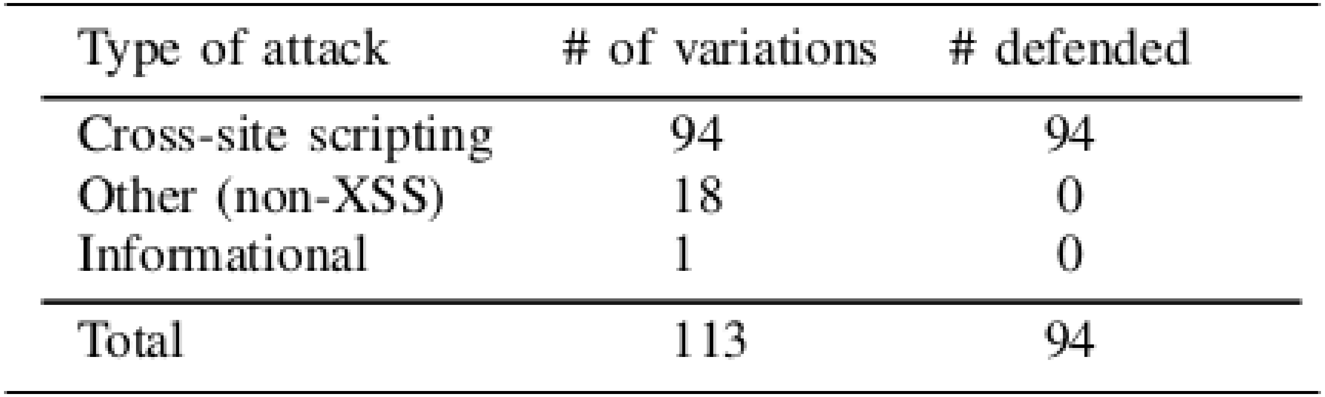


Context	Description	Example
cxAttrib	Element attribute	<td **align="center" nowrap**> ... </td>
cxAttribVal	Element attribute value	<a href="...**/untrusted.html**"> ... </a>
cxCData	Character data (CDATA)	<![CDATA[**untrusted**]]>
cxJSNumber	JavaScript numeric literal	var x = **10**;
cxJSString	JavaScript string literal	var x = "**untrusted**";
cxPCData	Parsed character data (PCDATA)	<p> **untrusted <i>content</i>** </p>
cxTitle	Document title	<title> Profile for user: **untrusted** </title>

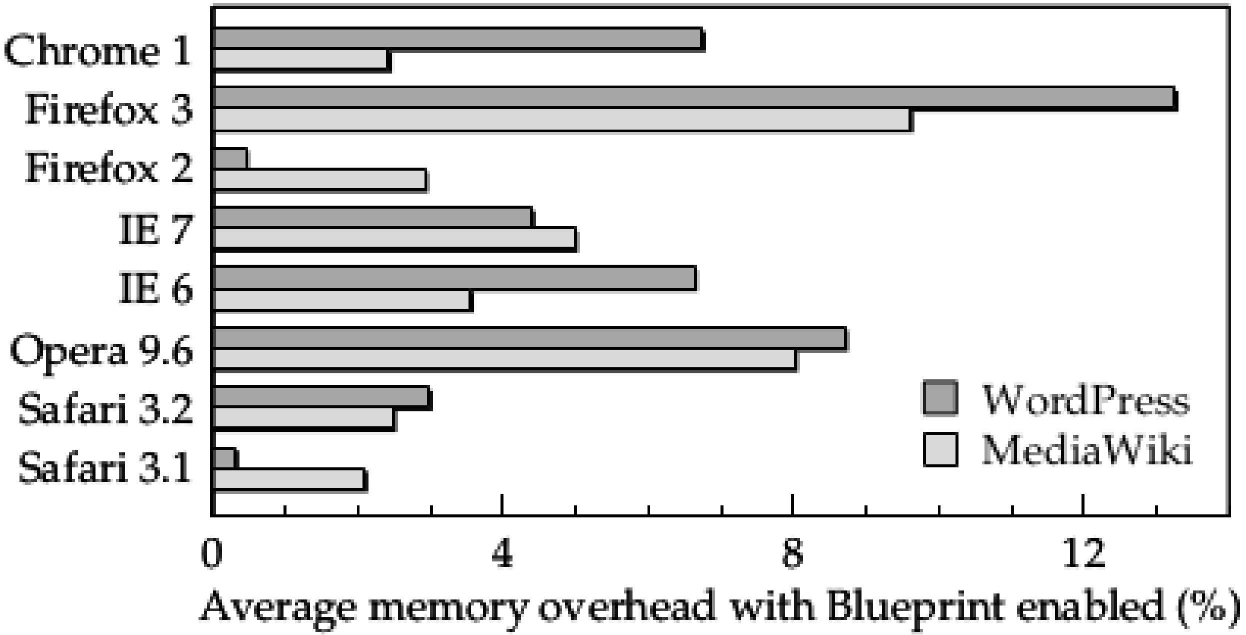
Evaluation

- 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

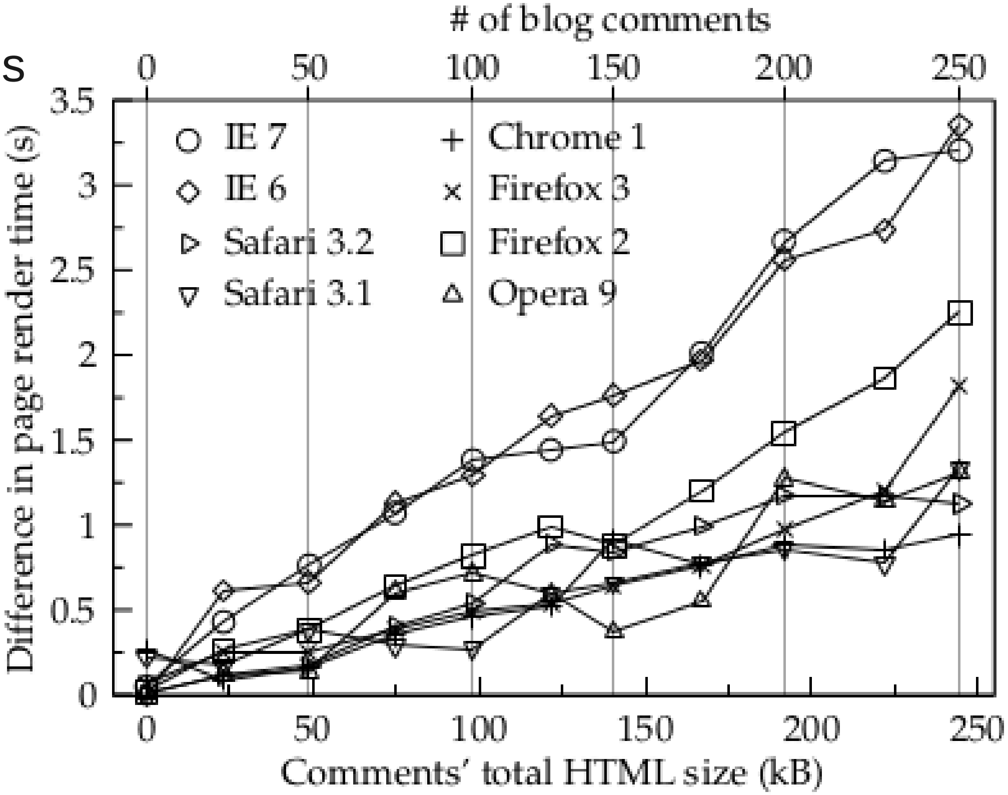


Type of attack	# of variations	# defended
Cross-site scripting	94	94
Other (non-XSS)	18	0
Informational	1	0
Total	113	94

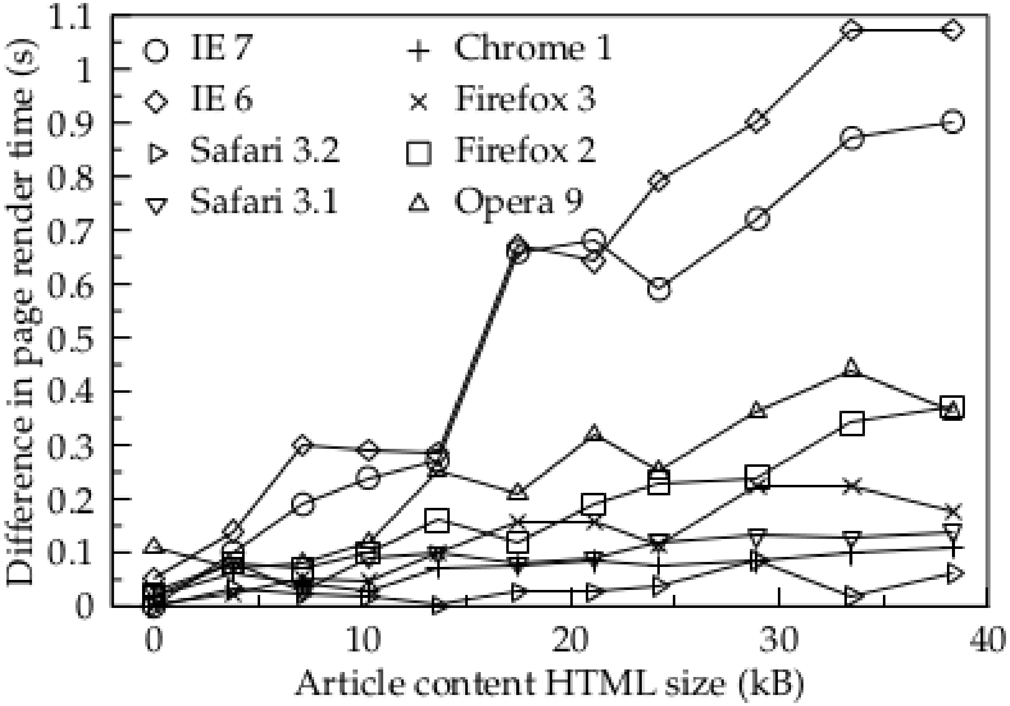


Results

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

- Ter Louw, Mike, and V. N. Venkatakrishnan. "Blueprint: Robust prevention of cross-site scripting attacks for existing browsers." Security and Privacy, 2009 30th IEEE Symposium on. IEEE, 2009.