Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks

Matthew Van Gundy and Hao Chen
University of California, Davis

16th Annual Network & Distributed System Security Symposium
Cross-Site Scripting (XSS) Vulnerabilities

\[\text{\texttt{p \ class='\text{	extacutenComment}' \}} \\text{'$\text{\textacutenComment}$} \\text{\texttt{\textbackslash n}}\text{\textbackslash /p}}\]
Great Article!

<p class='comment'>Great Article!</p>
Cross-Site Scripting (XSS) Vulnerabilities

<p class='comment'><script>p0wn()</script></p>
Cross-Site Scripting (XSS) Vulnerabilities

```
<p class='comment'>
  p0wn()</p>
</script>
```
Threat Model

- An attacker can submit arbitrary content to XSS-vulnerable applications
- An attacker cannot compromise web server or browser directly
- Malicious content must contain XHTML tags and attributes
Limitations of Existing Solutions

Server-side

▶ Server *sanitizes* untrusted data before sending it to the client
▶ Client may interpret data in an unexpected way
▶ E.g. Server replaces "\(<\text{script}>\)" with ""
  But attacker injects \(<\text{script}/xss>\)

Client-side

▶ Client enforces a server-specified policy

Challenges

▶ The client must know whether to trust content
▶ Attacker must not be able to forge trust metadata
Noncespaces Architecture

- Server partitions content into *trust classes*
- Server randomizes document to prevent forging of trust classification
- Server specifies policy of content permitted for each trust class
- Client displays the document only if it conforms to the policy
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = ("http://www.w3.org/1999/xhtml", "q")
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = ("http://www.w3.org/1999/xhtml", "q")
- FAQML question = ("urn:FAQML", "q")
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = ("http://www.w3.org/1999/xhtml", "q")
- FAQML question = ("urn:FAQML", "q")
- `<x : q xmlns:x = "http://www.w3.org/1999/xhtml">`
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = ("http://www.w3.org/1999/xhtml", "q")
- FAQML question = ("urn:FAQML", "q")
- `<x : q xmlns:x = "http://www.w3.org/1999/xhtml">`

NamespaceURI
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = ("http://www.w3.org/1999/xhtml", "q")
- FAQML question = ("urn:FAQML", "q")
- `<x : q xmlns:x="http://www.w3.org/1999/xhtml">`
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = ("http://www.w3.org/1999/xhtml", "q")
- FAQML question = ("urn:FAQML", "q")
- `<x : q xmlns:x = "http://www.w3.org/1999/xhtml">`
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`http://www.w3.org/1999/xhtml", "q")
- FAQML question = ("urn:FAQML", "q")
- `<x : q xmlns:x = "http://www.w3.org/1999/xhtml">`
- `<f:q xmlns:f="urn:FAQML">`

Noncespaces NDSS '09
Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = ("http://www.w3.org/1999/xhtml", "q")
- FAQML question = ("urn:FAQML", "q")

- `<x : q xmlns:x = "http://www.w3.org/1999/xhtml"`
  - `x`: prefix
  - `q`: name
  - `xmlns:x`: NamespaceURI

- `<f:q xmlns:f = "urn:FAQML"`

- `<faq:q xmlns:faq = "urn:FAQML"`
Defeating Node Splitting

▶ <x:a>...</x:a>
Defeating Node Splitting

- `<x:a>...</x:a>`
- `<x:a>...</a>`
Defeating Node Splitting

- `<x:a>...`</x:a>
- `<x:a>...`</a>
- `<x:a>...`</y:a>`
- `<x:a>...`</y:a>`
Encoding Trust Classifications

- Trusted

Noncespaces NDSS '09
Encoding Trust Classifications

- Trusted $\langle a \rangle \Rightarrow \langle t : a \rangle$
Encoding Trust Classifications

- Trusted $\langle a \rangle \Rightarrow \langle t : a \rangle$
- Untrusted $\langle a \rangle$

Noncespaces NDSS '09
Encoding Trust Classifications

- Trusted $<a> \Rightarrow <t:a>$
- Untrusted $<a>$
- Randomly choose trusted prefixes to prevent forgery
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>nile.com : ++Shopping</title>
</head>
<body>
<h1 id="title">{$item->name}</h1>
<h2>Reviews</h2>
<p class='review'>
  {$review}
</p>
</body>
</html>
Node Splitting Attack After Noncespaces

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:r617="http://www.w3.org/1999/xhtml">
<head>
    <title>nile.com : ++Shopping</title>
</head>
<body>
<h1 id="title">Useless Do-dad</h1>
<h2>Reviews</h2>
<p class="review">
    <script>p0wn()</script>
</p>
</body>
</html>
XSS Attack After Noncespaces

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<r617:html xmlns="http://www.w3.org/1999/xhtml"
xmlns:r617="http://www.w3.org/1999/xhtml">
<r617:head>
  <r617:title>nile.com : ++Shopping</r617:title>
</r617:head>
<r617:body>
<r617:h1 r617:id="title">Useless Do-dad</r617:h1>

<r617:h2>Reviews</r617:h2>
<r617:p r617:class='review'>
  <script src='http://badguy.com/p0wn.js' />
</r617:p>
</r617:body>
</r617:html>
Need for a client-side policy

Innocuous Input

<b>WARNING:</b>

</b>WARNING:</b>
Need for a client-side policy

Innocuous Input

<b>WARNING:</b> very important

<em>very</em> important
Need for a client-side policy

Innocuous Input

<b>WARNING:</b>

<em>very</em> important

Need for a client-side policy

Innocuous Input

<b>WARNING:</b>
<em>very</em> important

Malicious Input

<b onmouseover='...'>WARNING:</b>
Need for a client-side policy

Innocuous Input

<b>WARNING:</b><em>very</em> important

Malicious Input

<b onmouseover='...'>WARNING:</b><em onclick='...'>very</em> important
Need for a client-side policy

Innocuous Input

<b>WARNING:</b> very important

Malicious Input

<b onmouseover='...'>WARNING:</b>very important

Need for a client-side policy

XHTML

Policy

<b>
<em>
<a href='http://...'>
</a>
</em>
</b>

Noncespaces NDSS '09
Need for a client-side policy

**XHTML**

- `<b>`
- `<em>`
- `<a href='http://...'>`

**Policy**

- `allow //untrusted:b`
- `deny //@untrusted:onmouseover`
- `deny //@untrusted:*`
- `deny //@untrusted:href[starts-with(normalize-space(.), "javascript:")]`

Noncespaces NDSS '09
Need for a client-side policy

XHTML

Policy

\[ \text{allow} \ //\text{untrusted}:b \]

\[ \text{allow} \ //\text{untrusted}:em \]

\[ \text{allow} \ //\text{untrusted}:@\text{untrusted}:href[starts-with(normalize-space(.),"http:")]}] \]

\[ \text{deny} \ //\text{untrusted}:@\text{onmouseover} \]

\[ \text{deny} \ //\text{untrusted}:* \]

\[ \text{deny} \ //\text{untrusted}:@\text{href[starts-with(normalize-space(.),"javascript:")]}] \]
# Need for a client-side policy

<table>
<thead>
<tr>
<th>XHTML</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;b&gt;</code></td>
<td><code>allow //untrusted:b</code></td>
</tr>
<tr>
<td><code>&lt;em&gt;</code></td>
<td><code>allow //untrusted:em</code></td>
</tr>
<tr>
<td><code>&lt;a href='http:...'&gt;</code></td>
<td><code>allow //untrusted:a/@untrusted:href[starts-with(normalize-space(.), &quot;http:&quot;)]</code></td>
</tr>
</tbody>
</table>

Noncespaces NDSS '09
Need for a client-side policy

XHTML

<b>
allow //untrusted:b
</b>

<em>
allow //untrusted:em
</em>

<a href='http:...'>
allow //untrusted:a/@untrusted:href[starts-with(normalize-space(.), "http:")]</a>

<b onmouseover=''></b>

<em onclick=''></em>

<a href='java...'></a>

Policy

allow //untrusted:b

allow //untrusted:em

allow //untrusted:a/@untrusted:href[starts-with(normalize-space(.), "http:")]
Need for a client-side policy

**XHTML**

```html
<br>
<em>
<a href='http:...'>
</a>
<br onmouseover=''
<em onclick=''
<a href='java...'>
```

**Policy**

```xml
allow //untrusted:b
allow //untrusted:em
allow //untrusted:a/@untrusted:href[
    starts-with(normalize-space(.),
    "http:")]
```
```xml
<onmouseover=''
<onclick=''
```
```xml
deny //@untrusted:onclick
```
Need for a client-side policy

**XHTML**

```xml
<b>
<em>
<a href='http:...'>
</a>
</em>
</b>
```

**Policy**

```
allow //untrusted:b
allow //untrusted:em
allow //untrusted:a/@untrusted:href[
  starts-with(normalize-space(.),
  "http:")]

deny //@untrusted:onmouseover
deny //@untrusted:*
```

Noncespaces NDSS '09
Need for a client-side policy

**XHTML**

```html
<b>
allow //untrusted:b
</b>

```html
<em>
allow //untrusted:em
</em>

```html
<a href='http:...'>
allow //untrusted:a/@untrusted:href[
    starts-with(normalize-space(.),
    "http:"])
</a>

```html
<b onclick=''>
deny//@untrusted:onclick
</b>

```html
<em onclick=''>
deny//@untrusted:*
</em>

```html
<a href='java...'>
deny//@untrusted:href[
    starts-with(normalize-space(.),
    "javascript:"))
</a>
```

Noncespaces NDSS '09
Determining Trusted Content

- Design patterns separate presentation and business logic
- Templates contain static HTML (presentation)

- Program creates dynamic content from user input
Determining Trusted Content

- Design patterns separate presentation and business logic
- Templates contain static HTML (presentation)
  - Classify as trusted
- Program creates dynamic content from user input
Determining Trusted Content

- Design patterns separate presentation and business logic
- Templates contain static HTML (presentation)
  - Classify as trusted
- Program creates dynamic content from user input
  - Classify as untrusted
Modifications to Smarty

Application Logic

Smarty Engine

Randomization Preprocessor

Smarty Compiler

Noncespaces

NDSS '09
Modifications to Smarty

{$foo}

Template

Smarty Engine

Randomization Preprocessor

Smarty Compiler

Application Logic
Modifications to Smarty

{${foo}$}  →  Smarty Engine  →  Smarty Compiler

Randomization Preprocessor

Template  →  Application Logic
Modifications to Smarty

Application Logic

Smarty Engine

Randomization Preprocessor

Smarty Compiler

Template

PHP Code
Modifications to Smarty

Application Logic

Values for template variables

Template

Smarty Engine

Randomization Preprocessor

Smarty Compiler

PHP Code
Modifications to Smarty

Application Logic

Values for template variables

Template

Smarty Engine

XHTML Doc

Randomization Preprocessor

Smarty Compiler

PHP Code

Noncespaces NDSS ’09
Modifications to Smarty

{${foo}$} → Template → Smarty Engine → Application Logic

Randomization Preprocessor
Smarty Compiler
Modifications to Smarty

Application Logic

Template

Smarty Engine

Template w/ static XML namespace prefixes

Randomization Preprocessor

Smarty Compiler
Modifications to Smarty

Modifications to Smarty

{${foo}$} $\rightarrow$ Template $\rightarrow$ Smarty Engine

Template w/ static XML namespace prefixes

Randomization Preprocessor $\rightarrow$ Smarty Compiler

XML namespace prefixes randomized

Application Logic
Modifications to Smarty

Application Logic

{${foo}$} → Template → Smarty Engine

Smarty Engine:
- Template with static XML namespace prefixes
- PHP Code
- Smarty Compiler
- Randomization Preprocessor

Randomization Preprocessor → XML namespace prefixes randomized
Modifications to Smarty

Application Logic

Values for template variables

Template

Smarty Engine

Template w/ static XML namespace prefixes

Randomization Preprocessor

PHP Code

Smarty Compiler

XML namespace prefixes randomized

Noncespaces NDSS '09
Modifications to Smarty

Application Logic

Values for template variables

Smarty Engine

Randomized XHTML Doc

PHP Code

Smarty Compiler

Randomization Preprocessor

Template w/ static XML namespace prefixes

Template

{$foo}
Client-side Modifications
Client-side Modifications
Client-side Modifications
Client-side Modifications
Client-side Modifications

Noncespaces NDSS '09
Client-side Modifications

GET /

GET /policy

Proxy
Client-side Modifications

Noncespaces

NDSS '09
Evaluation

- Tested effectiveness of Noncespaces on 2 applications
- Developed policy for each application
- Ensured that Noncespaces stopped a number of XSS attacks
- Measured performance overhead of both server-side randomization and client-side policy checking
Evaluation

- Baseline
- Server randomization w/o proxy
- Server randomization w/ proxy

**# of concurrent requests**

**Avg. Requests/sec**

- 0
- 0.1
- 0.2
- 0.3
- 0.4
- 0.5
- 0.6
- 0.7
- 0.8
- 1
- 10
- 30

Noncespaces NDSS '09
Evaluation

![Bar chart showing the comparison of different server randomization methods.
- Baseline
- Server randomization w/o proxy
- Server randomization w/ proxy

The chart compares the average requests per second against the number of concurrent requests. The Baseline method consistently shows the highest average requests per second, followed by Server randomization w/o proxy, and then Server randomization w/ proxy.

Noncespaces NDSS '09
Evaluation

Baseline
Server randomization w/o proxy
Server randomization w/ proxy

Avg. Requests/sec
# of concurrent requests

Noncespaces NDSS '09
Evaluation

Noncespaces NDSS '09
Evaluation

Baseline
Server randomization w/o proxy
Server randomization w/ proxy

0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1
10
30

Avg. Requests/sec
# of concurrent requests

Noncespaces NDSS '09
Evaluation

- Baseline
- Server randomization w/o proxy
- Server randomization w/ proxy

Bar chart showing Avg. Requests/sec vs. # of concurrent requests.
Evaluation

- Baseline
- Server randomization w/o proxy
- Server randomization w/ proxy

![Graph showing average requests per second vs. number of concurrent requests]

Noncespaces NDSS '09
Evaluation

Baseline
Server randomization w/o proxy
Server randomization w/ proxy

Noncespaces NDSS '09
Evaluation

![Bar chart showing the performance of different server randomization methods.](chart.png)

- **Baseline**
- **Server randomization w/o proxy**
- **Server randomization w/ proxy**

The chart compares the average requests per second (Avg. Requests/sec) for different numbers of concurrent requests (1, 10, and 30) across the baseline and server randomization methods. The results indicate that server randomization with a proxy generally performs better than the baseline method, especially under higher loads.
Related Work

- Instruction Set Randomization (Kc et al., CCS ’03) and (Barrantes et al., CCS ’03)
- BEEP (Jim et al., WWW ’07)
- Mutation Event Transforms (Erlingsson et al., HotOS ’07)
- Noxes (Kirda et al., ACM SAC ’06)
- Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis (Vogt et al., NDSS ’07)
Conclusion

- We can achieve security without data sanitization on the server
  - Servers classify how trustworthy content is
  -Servers convey trust classifications in a tamper resistant way
  -Clients interpreting the content enforce the policy
- Leverage randomization and XML features to thwart XSS attacks
- Leverage design paradigms to determine trust information without dynamic information flow tracking
Questions?
Example Noncespaces Policy

1 namespace trusted
2 namespace untrusted

4 allow //trusted:* 
5 allow //trusted:@*

7 allow //untrusted:b 
8 allow //untrusted:i 
9 allow //untrusted:u 
10 allow //untrusted:a 
11 allow //untrusted:a/@untrusted:href[
12 starts-with(normalize-space(.), "http:" )]
13 allow //untrusted:img 
14 allow //untrusted:img/@untrusted:src[
15 starts-with(normalize-space(.), "http:" )]

17 deny //*
18 deny //@*