Document Structure Integrity: A Robust Basis for Cross-Site Scripting Defense

Yacin Nadji
Illinois Institute Of Technology

Prateek Saxena
UC Berkeley

Dawn Song
UC Berkeley
A Cross-Site Scripting Attack

Hi Joe,
<img src="...">
<script src=""></script>

Cookies, Password

Policy: ALLOW
{a, a@href, img, img@src }

Hi Joe,
<img src="...">
<script src=""></script>
Limitations of Server-side Sanitization

- `<IMG SRC="javascript:alert('XSS')">`
- `<IMG SRC=JavaScRiPt:alert('XSS')>`
- `<IMG SRC=&#106;&#97;&#118;&#97; &#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;& #116;&#40;&#39; &#88;&#83;&#83;&#39;&#41;>`

Policy: ALLOW

{a, a[href, img, img[src] }
Limitations of Server-side Sanitization

- Over 90 ways to inject JS [RSnake07]

- Multiple Languages
  - JS, Flash, CSS, XUL, VBScript

Cookies, Password

Hi Joe,

<img src=...>
A Different Approach…

- Previous defenses: XSS is a sanitization problem
- Our view: XSS is a document structure integrity problem
Concept of Document Structure

STATIC DOCUMENT STRUCTURE

DYNAMIC DOCUMENT STRUCTURE

JAVASCRIPT

document.write()
Document Structure Integrity (DSI)

• **Definition:**
  – Given a server’s policy P,
  – Restrict untrusted content to allowable syntactic elements
  – Policy in terms of client-side languages

• **Central idea for DSI enforcement**
  – Dynamic information flow tracking (server & browser)
  – Policy based parser-level confinement

• **Default policy:** Only leaf nodes untrusted
Talk Outline

• Power of DSI Defense: Examples
• Design Goals
• Architecture
• Implementation
• Evaluation
• Conclusion & Related Work
Talk Outline

- Power of DSI Defense: Examples
- Design Goals
- Architecture
- Implementation
- Evaluation
- Conclusion & Related Work
DSI Defense: A Powerful Approach

• DSI enforcement prevents
  – Not just cookie-theft
    » Form injection for phishing [Netcraft08]
    » Profile Worms [Samy05, Yammaner06]
    » Web site defacement through XSS
  – “DOM-Based” XSS (Attacks on client-side languages)
  – Vulnerabilities due to browser-server inconsistency
Example 1: DOM-Based XSS

- DOM-based client-side XSS [Klein05]

```html
<div id="Joe; online">
  <div id="Joe">online+</div>
</div>
```

JAVASCRIPT  DYNAMIC UPDATE
Example 1: DOM-Based XSS

- DOM-based client-side XSS [Klein05]

```html
<div id="Devil; <script>..</script>">
  <!-- JAVASCRIPT content here -->
</div>
```
Example 1: DOM-Based XSS

- DOM-based client-side XSS [Klein05]

```html
<div id="Devil; <script>.. </script>">
<script>
</script>
</div>
```

```
<iframe src="javascript:alert("Devil")"></iframe>
```

```
<iframe src="javascript:alert("..")"></iframe>
```

DYNAMIC UPDATE

JAVASCRIPT

script

“Devil”

“..”
Example 2: Inconsistency Bugs

1. **Browser-Server Inconsistency Bugs**

Assumed Parse Tree

```
<img onload=alert(1)>
```

```
<img onload:=alert(1)>
```

```
<img onload:=alert(1)>
```
Talk Outline

• Defense in Depth: Examples
• **Design Goals**
• Architecture
• Implementation
• Evaluation
• Conclusion & Related Work
Design Goals

- Clear separation between policy and mechanism
- No dependence on sanitization
- No changes to web application code
- Minimize false positives
- Minimizes impact to backwards compatibility
- Robustness
  - Address static & dynamic integrity attacks
  - Defeat adaptive adversaries
Mechanisms

• Client-server architecture

• Server
  – Step 1: Identify trust boundaries in HTML response
  – Step 2: Serialize
    » Encoding data & trust boundaries in HTML

• Client
  – Step 3: De-serialize
    » Initialize HTTP response page into static document structure
  – Step 4: Dynamic information flow tracking
    » Modified semantics of client-side interpretation
Talk Outline

- Defense in Depth: Examples
- Design Goals
- Architecture
- Implementation
- Evaluation
- Conclusion & Related Work
Approach Overview: Static DSI

SERVER

SERIALIZER

DE-SERIALIZER

BROWSER

<img src="...">
<script src=""></script>
<a href="..."></a>

[<img src="...">
<script src=""></script>]
<a href="..."></a>

img
script

P
Approach Overview: Dynamic DSI

SERVER

<div id="Devil; <script>..</script>"/>

DE-SERIALIZER

<div id="[[Devil; <script>..</script>]]"/>

TAINT TRACKING

BROWSER

<div id="id" Devil;<script>..<script>"/>
Approach Overview: Dynamic DSI (II)

SERVER

SERIALIZER

<div id="Devil; <script>..</script>">

DE-SERIALIZER

<div id="[[Devil; <script>..</script>]]">

TAINT TRACKING

BROWSER

id

“Devil”

script

.."
Serialization Design: Key Challenge

- Safety against an adaptive adversary
Serialization: Key Challenge

- Do not rely on sanitization

```html
<CONFINE ... ID="N5"></CONFINE>
<SCRIPT>
document.getElementById("N5").innerHTML = ""
</SCRIPT>
```

What to disallow?
Serialization Design: Key Challenge

- Attack on sanitization mechanism for JS strings

```html
<CONFINE ... ID="N5"></CONFINE>
<SCRIPT>
    document.getElementById("N5").innerHTML = ""
</SCRIPT>

<SCRIPT>
    Attack
</SCRIPT>

```

```html
';
</SCRIPT>
```
Markup Randomization

- Markup Randomization
  - Mechanism independent of the policy
  - Does not depend on any sanitization

Valid Nonces: 00101, 11010, 01110
Policy: ALLOW {a, a@aref ... }
Markup Randomization

- Markup Randomization
  - Mechanism independent of the policy
  - Does not depend on any sanitization

Valid Nonces: **00101, 11010, 01110**
Policy: ALLOW \{a, a@aref\}

\[[00101 \mathbf{R}]_{00101}\] \rightarrow \[[00101 \mathbf{R}]_{00101}\]

OK!
Markup Randomization

- Markup Randomization
  - Mechanism independent of the policy
  - Does not depend on any sanitization

Valid Nonces: 00101, 11010, 01110
Policy: ALLOW \{a, a@aref\}
Browser-side Taint Tracking

• Dynamic DSI

• Client Language Interpreters enhanced

• Ubiquitous tracking of untrusted data in the browser
Talk Outline

• Advantages of DSI in Attack Coverage
• Design Goals
• Architecture
• Implementation
• Evaluation
• Conclusion & Related Work
Implementation

• Full Prototype Implementation

• DSI-enable server
  – Utilized existing taint tracking in PHP [IBM07]

• DSI-compliant browser
  – Implemented in KDE Konqueror 3.5.9
  – Client side taint tracking in JS interpreter of KDE 3.5.9
You are Owned!
In a DSI-compliant Browser…

```html
<script>alert(document.cookie)</script>
```
Talk Outline

• Advantages of DSI in Attack Coverage
• Design Goals
• Architecture
• Implementation
• **Evaluation**
• Conclusion & Related Work
Evaluation: Attack Detection

• Stored XSS attacks

• Vulnerable phpBB forum application
• 25 public attack vectors [RSnake07]
• 30 benign posts

• Results
  – 100% attack prevention
  – No changes required to the application
  – No false positives
Evaluation: Real-World XSS Attacks

• 5,328 real-world vulnerabilities [xssed.com]
• 500 most popular benign web sites [alexa.com]
• Default Policy:
  – Coerce untrusted data to leaf nodes
• Results
  – 98.4% attack prevention
  – False Negatives:
    » Due to exact string matching in instrumentation
  – False Positives: 1%
    » Due to instrumentation for tainting (<title> on Slashdot)
## Evaluation: Performance

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Browser Overhead</strong></td>
<td>1.8%</td>
</tr>
<tr>
<td><strong>Server overhead</strong></td>
<td>1-3%</td>
</tr>
<tr>
<td><strong>Static page size increase</strong></td>
<td>1.1%</td>
</tr>
</tbody>
</table>
Related Work

- **Client-server Approaches**
  - BEEP [Jim07]
  - `<jail>` [Eich07]
  - Hypertext Isolation [Louw08]

- **Client-side approaches**
  - IE 8 Beta XSS Filter [IE8Blog]
  - Client-side Firewalls [Kirda06]
  - Sensitive Info. Flow Tracking [Vogt07]

- **Server-side approaches**
  - Server-side taint-based defenses [Xu06, Nan07, Ngu05, Pie04]
  - XSS-Guard [Bisht08]
  - Program Analysis for XSS vulnerabilities [Balz08, Mar05, Mar08, Jov06, Hua04]
Conclusion

• DSI: A fundamental integrity property for web applications
• XSS as a DSI violation
• Multifaceted Approach
  – Clearly separates mechanism and policy
• Defeats adaptive adversaries
  – Markup randomization
• Evaluation on a large real-world dataset
  – Low performance overhead
  – No web application code changes
  – No false positives with configurable policies
Questions

Thank you!
Client-Side Proxy

www.site.com?user=Joe

user=Joe

Hi [[Joe]]!

Hi Joe!
Markup Randomization: Adaptive Attacks

- Multiple valid parse trees

```
```

OR

```
```
Attack Coverage (II): Inconsistency Bugs

• Browser-Server Inconsistency Bugs