Preventing JavaScript Injection Attacks

Definition: Attacks can be done whenever information is gathered from a user and redisplayed to the screen. A feedback form allows a prime example of this. Most feedback allow 200 to 500 characters which gives an intruder the ability to write considerable JavaScript that will be rendered the next time the page is posted.

Example of JavaScript Injection attacks:

- Cross-Site Scripting (XSS) – Sending of valuable information from the attacked site to another site (IE. Gathering cookies, passwords, CC numbers etc).

Protection of JavaScript Injection

APPROACH #1: HTML ENCODE

- Html.Encode replaces dangerous characters like '<' with a reference variable &lt;
- Feedback.Message is encoded prior to being displayed to the screen
- Only encode information entered by the user or displayed back to screen from database.

  Encode the results of information intended to be re-displayed to the screen.
  <%=Html.Encode (feedback.Message) %>

  Example_Code_01

  Attack: <script>alert("Boo!")</script>
  - Becomes -
  Converted: &lt;script&gt;alert(&quot;Boo!&quot;)&lt;/script&gt;.
  - Renders -
  Displays: <script>alert("Boo!")</script>

APPROACH #2: HTML ENCODE IN THE CONTROLLER

- Html.Encode the input prior to adding it to the database.

  Example_Code_02

```
public ActionResult Create(string message)  {
  // Add feedback
  var newFeedback = new Feedback();
  newFeedback.Message = Server.HtmlEncode(message);
  newFeedback.EntryDate = DateTime.Now;
  db.Feedbacks.InsertOnSubmit(newFeedback);
  db.SubmitChanges();
  // Redirect
  return RedirectToAction("Index");
}// End action Create
```

** **NOTE:** Approach #2 leaves ugly data in your database as it is encoded prior to saving in order to prevent the injection. Attempting to display the information from the database in any other medium will cause an issue.