**Preventing JavaScript Injection Attacks**

이 튜토리얼의 목적은 사용자가 ASP.NET MVC 어플리케이션에서 어떻게 자바스크립트 인젝션 공격을 막는가 설명하는 것입니다. 이 튜토리얼은 자바스크립트 인젝션 공격을 막는 두 가지 방법에 대해 토의할 것입니다. 화면에 보여줄 데이터를 인코딩 함으로써 어떻게 자바스크립트 인젝션 공격을 막는지 알게 될 것입니다.

**What is a JavaScript Injection Attack?**

사용자의 입력을 받고 입력에 의해 화면이 다시 보여질 때마다, 웹 사이트는 자바스크립트 인젝션 공격에 오픈 됩니다. 자바스크립트 인젝션 공격에 오픈 된 실제 어플리케이션을 시험해보겠습니다.

아래 사진과 같은 사용자 입력 웹 사이트를 만들었다고 가정합니다. 고객은 사이트에 접속하고, 제품을 사용하고 느낀 경험에 대해 의견을 입력할 수 있습니다. 고객이 그들의 의견을 전송할 때, 의견은 의견 페이지로 다시 보여질 것입니다.



고객 의견 웹 사이트는 아래 코드의 controller을 사용합니다. 이 controller은 Index()와 Create() 라는 두 actions을 가지고 있습니다.

|  |
| --- |
| using System;using System.Web.Mvc;using CustomerFeedback.Models;namespace CustomerFeedback.Controllers{ [HandleError] public class HomeController : Controller { private FeedbackDataContext db = new FeedbackDataContext(); public ActionResult Index() { return View(db.Feedbacks); } public ActionResult Create(string message) { // Add feedback var newFeedback = new Feedback(); newFeedback.Message = message; newFeedback.EntryDate = DateTime.Now; db.Feedbacks.InsertOnSubmit(newFeedback); db.SubmitChanges(); // Redirect return RedirectToAction("Index"); } }} |

Index() 메소드는 Index view을 보여줄 것입니다. 이 메소드는 데이터베이스로부터 의견을 가져와 Index view 로 고객 의견을 전달합니다.

Create() 메소드는 새로운 Feedback 개체를 만들고 그것을 데이터베이스에 저장합니다. 고객이 폼에서 입력한 메시지는 message 파라메터로써 Create() 메소드에 전달됩니다. Feedback 개체가 생성되고, message은 생성된 feedback 개체의 Message 속성에 값이 입력됩니다. Feedback 개체는 DataContext.SubmitChanges() 호출을 통해 데이터베이스에 전송됩니다. 마지막으로 모든 의견이 보여지는 Index view 페이지로 리다이렉트 됩니다.

Index view은 다음 코드를 가지고 있습니다.

|  |
| --- |
| <%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="CustomerFeedback.Views.Home.Index"%><%@ Import Namespace="CustomerFeedback.Models" %><asp:Content ID="indexContent" ContentPlaceHolderID="MainContent" runat="server"> <h1>Customer Feedback</h1> <p> Please use the following form to enter feedback about our product. </p> <form method="post" action="/Home/Create"> <label for="message">Message:</label> <br /> <textarea name="message" cols="50" rows="2"></textarea> <br /><br /> <input type="submit" value="Submit Feedback" /> </form> <% foreach (Feedback feedback in ViewData.Model) {%> <p> <%=feedback.EntryDate.ToShortTimeString()%> -- <%=feedback.Message%> </p> <% }%></asp:Content> |

Index view은 두 가지 영역으로 나뉩니다. 상단은 고객 의견 폼이고, 하단은 이전 저장된 고객 의견 아이템을 보여줄 For..Each 루프 영역입니다.

고객 의견 웹 사이트는 간단한 사이트로, 불행히도 자바스크립트 인젝션 공격에 노출되어 있습니다.

고객 의견 폼에서 다음과 같은 문자를 입력한다고 생각해보세요.

<script>alert(“Boo!”)</script>

이 문자는 경고 메시지 박스를 띄우는 자바스크립트입니다. 누군가 의견 폼에 이 스크립트를 입력하고 전송을 누르면 앞으로 누구든지 이 사이트에 방문하게 되면 메시지 박스 Boo! 가 나타날 것입니다.



자바스크립트 인젝션 공격에 대한 당신의 초기 반응은 무관심일지도 모릅니다. 아마 자바스크립트 인젝션 공격은 화면을 공격하는 단순한 유형으로 생각할 것입니다. 자바스크립트 인젝션 공격으로 실제로 어떠한 것도 할 수 없다고 생각해보세요.

불행이 해커는 실제로 웹 사이트에 자바스크립트 인젝션으로 사이트를 마비시킬 수 있습니다. Cross-Site Scripting (XSS) 공격을 하기 위해 자바스크립트 인젝션을 사용할 수 있습니다. Cross-Site Scripting 공격으로, 사용자의 정보를 훔칠 수 있고, 다른 웹 사이트로 정보를 보낼 수도 있습니다.

예를 들어, 해커는 다른 유저로부터 브라우저 쿠키 값을 훔칠 수도 있습니다. 비밀번호, 주민등록번호, 또는 신용 카드 번호와 같은 중요한 정보가 쿠키에 저장되어 있다면, 해커들은 그 정보를 알아낼 수 있습니다. 사용자가 자바스크립트 공격에 노출된 페이지에 있는 입력 폼에 정보를 입력한다면, 해커는 자바스크립트 인젝션 공격으로 정보를 가로채 다른 웹 사이트로 보낼 것입니다.

자바스크립트 인젝션 공격은 신중히 처리해야 하고, 사용자의 중요한 정보는 보호해야 할 것입니다. 다음 두 영역에서, 당신이 자바스크립트 인젝션 공격으로부터 ASP.NET MVC 어플리케이션을 보호할 수 있는 두 기술을 이야기 할 것입니다.

**Approach #1: HTML Encode in the View**

자바스크립트 인젝션 공격을 막는 쉬운 방법은 사용자가 입력한 데이터가 view에 보여질 때, 입력한 데이터를 HTML 인코딩 하는 것입니다. 다음 코드의 수정된 Index view을 보세요..

|  |
| --- |
| <%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="CustomerFeedback.Views.Home.Index"%><%@ Import Namespace="CustomerFeedback.Models" %><asp:Content ID="indexContent" ContentPlaceHolderID="MainContent" runat="server"> <h1>Customer Feedback</h1> <p> Please use the following form to enter feedback about our product. </p> <form method="post" action="/Home/Create"> <label for="message">Message:</label> <br /> <textarea name="message" cols="50" rows="2"></textarea> <br /><br /> <input type="submit" value="Submit Feedback" /> </form> <% foreach (Feedback feedback in ViewData.Model) {%> <p> <%=feedback.EntryDate.ToShortTimeString()%> -- <%=Html.Encode(feedback.Message)%> </p> <% }%></asp:Content> |

Feedback.Message의 값은 아래와 같은 코드로 값이 화면에 출력되기 전에 HTML 로 인코딩 되었습니다.

|  |
| --- |
| <%=Html.Encode(feedback.Message)%> |

HTML을 문자로 인코딩한다는 것은 무엇을 의미하는 것일까? HTML을 문자로 인코딩 할 때, 위험한 문자는, 예를 들어 < 와 >, HTML entity reference 로, 예를 들어 &lt; 와 &gt; 로 바뀝니다. 그래서, 문자열 <script>alert(“Boo!”)</script>가 HTML 인코딩 되면, &lt;script&gt;alert(&quot;Boo!&quot;)&lt;/script&gt;. 로 바뀌게 된다. 인코딩 된 문자는 더 이상 자바스크립트로 실행되지 않습니다.



**Approach #2: HTML Encode in the Controller**

view에서 데이터가 화면에 보여질 때, HTML 인코딩 된 데이터 대신, 데이터베이스에 데이터를 전송하기 전에 데이터를 HTML 인코딩 할 수도 있습니다. 두 번째 방안은 아래의 코드처럼 controller 의 경우입니다.

|  |
| --- |
| using System;using System.Web.Mvc;using CustomerFeedback.Models;namespace CustomerFeedback.Controllers{ [HandleError] public class HomeController : Controller { private FeedbackDataContext db = new FeedbackDataContext(); public ActionResult Index() { return View(db.Feedbacks); } 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"); } }} |

Message의 값이 Create() action 내에서 데이터베이스로 전송되기 전에 HTML 인코딩 된 것을 확인해야 합니다. Message가 view에 다시 보여질 때, Message은 HTML 인코딩 되고, 자바스크립트 인젝션도 실행되지 않을 것입니다.

전형적으로, 두 번째 접근 방식보다 첫 번째 접근 방식을 선호합니다. 두 번째 방식의 문제는 데이터베이스에 HTML 인코딩 된 데이터가 들어가는 것입니다. 다른 말로, 데이터베이스의 데이터가 보기에 좋지 않은 문자들로 가득할 것입니다.

이것이 왜 나쁜 것일까요? 웹 페이지에서 더 이상 이 데이터베이스의 데이터를 표시할 필요가 없을 경우 문제가 생길 수 있습니다. 예를 들어 윈도우 어플리케이션에서 사용할 때 입니다.