TÌM TRONG BLOG

SERIES HƯỚNG DẪN

Lên trên

DANH MỤC BÀI VIẾT

Lên trên

BÀI MỚI NHẤT

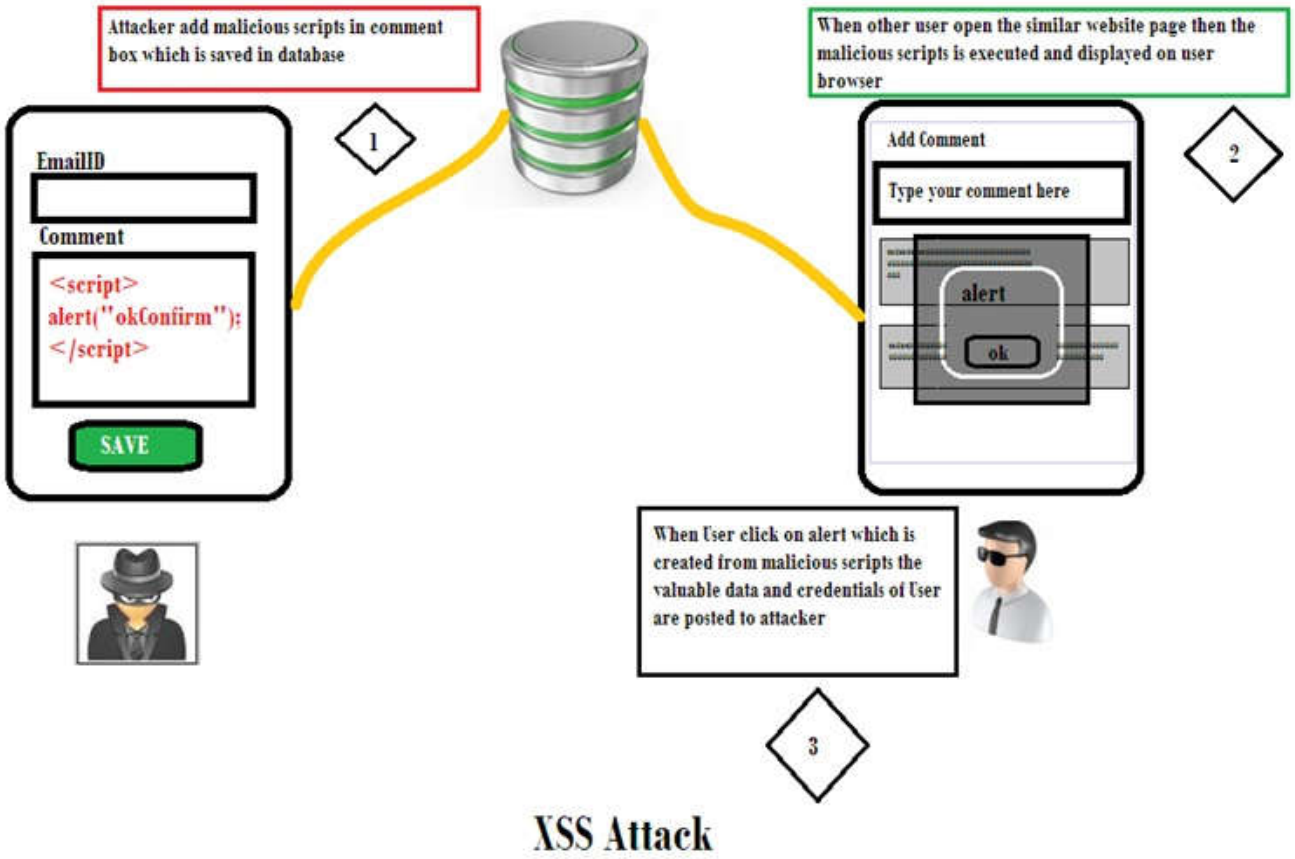
TAGS

Angular2 Căn Bản (/Tag/Angular2-Can-Ban.Html) Angular 2 Miễn Phí (/Tag/Angular-2-Mien-Phi.Html)

Series bảo mật trong ASP.NET MVC - 3: Tấn công Cross-Site Scripting (XSS)

vào **06/07/2017**. Lượt xem: **2,844**

Tấn công Cross-site Scripting (XSS) là cách tấn công bằng việc đẩy các đoạn mã javascript vào hệ thống thông qua các trường nhập liệu, kiểu tấn công này là phổ biến nhất và cho phép các hacker có thể lấy cắp thông tin đăng nhập và các dữ liệu có giá trị nhằm khai thác hệ thống.



Lên trên

Hình 1. Cross Site Scripting (XSS).

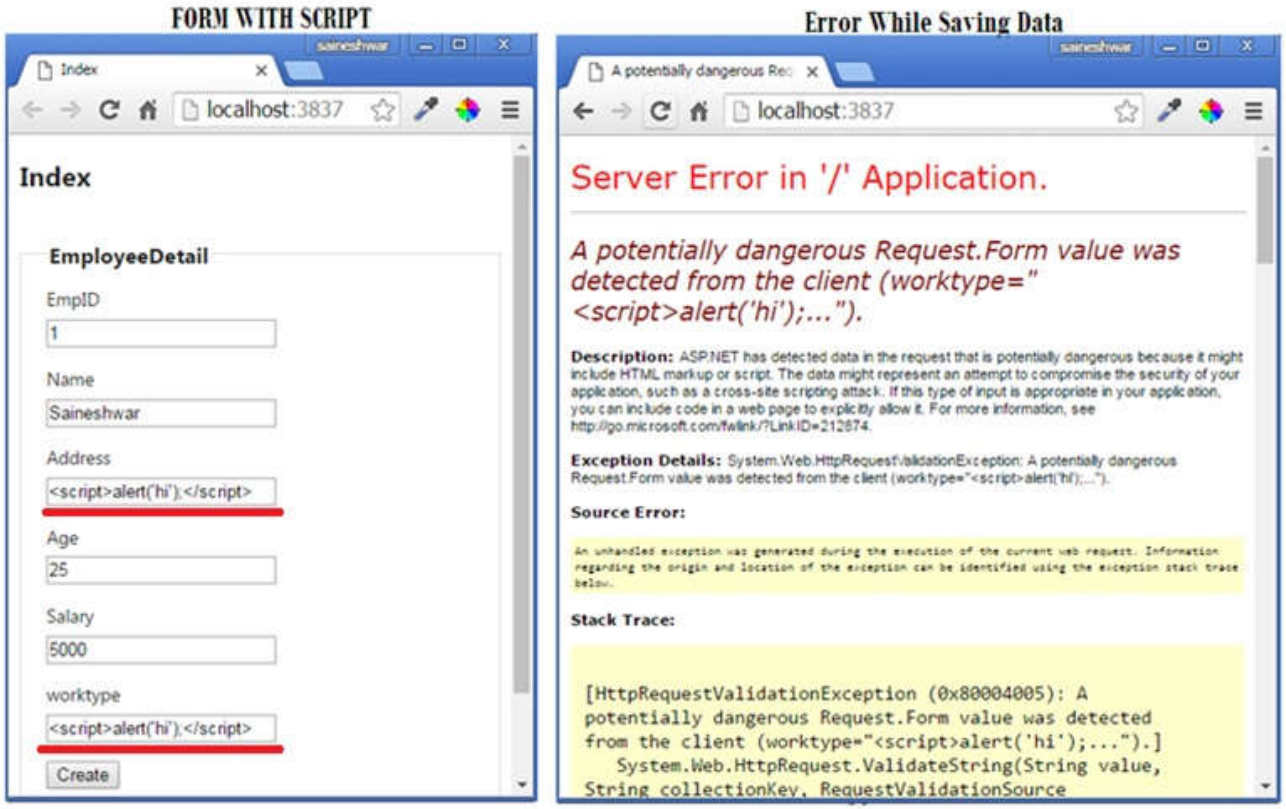
Hacker sẽ ghé thăm một website sau đó cố gửi các đoạn mã script có hại vào ô comment hoặc bất cứ ô nhập liệu nào. Nếu website không kiểm tra và cho phép nhúng các mã độc thì code sẽ được thực thi trên website của bạn.

Hãy thử ví dụ dưới đây một form nhân viên chúng ta dùng để lưu dữ liệu. Trong textbox tôi đã cố gắng thực thi một vài đoạn mã có hại sử dụng javascript sử dụng thẻ SCRIPT. Nhưng nếu chúng ta cố gắng thực thi chúng trên MVC sẽ gặp lỗi.

Mặc định ASP.NET đã ngăn chặn tấn công XSS

Một giá trị nguy hiểm được tìm thấy trong Request.Form như sau: ("<script>alert('hi');").

Đây là lỗi xẩy ra bởi vì MVC đã kiểm tra dữ liệu được nhập vào từ người dùng và nếu người dùng cố gắng thực thi đoạn script sẽ không được phép.



Hình 2. Cố tính gửi đoạn script có hại sẽ gặp lỗi

Lên trên

Nhưng nếu chúng ta cần đẩy thẻ SCRIPT lên. Ví dụ trang web lập trình đăng code mẫu sẽ cần người dùng gửi code mẫu lên có chứa thẻ script. Trong tình huống này chúng ta có 4 thứ có thể cho phép script được đăng lên:

Giải pháp: -

1. **[ValidateInput(false)]**
2. **[AllowHtml]**
3. **[RegularExpressionAttribute]**
4. **AntiXSS Library**

Giải pháp 1:

ValidateInput

[ValidateInput] là một attribute có thể áp dụng trên cả Controller hoặc Action Method mà chúng ta muốn script được đi qua.

Nếu chúng ta muốn cho phép đăng đoạn mã lên chỉ cần set thuộc tính thành False (**[ValidateInput(false)]**) chúng ta sẽ không validate nội dung truyền lên. Nếu chúng ta set thành true [**ValidateInput(true)]**) thì nó sẽ kiểm tra nội dung truyền lên. Theo cách này nếu bạn áp dụng nó vào Controller thì nó sẽ áp dụng vào toàn bộ các action method trong controller đó và nếu bạn chỉ áp dụng trên từng Action Method thì sẽ chỉ có tác dụng trên method đó mà thôi.

ValidateInput áp dụng cho tất cả các property của Model (EmployeeDetails).

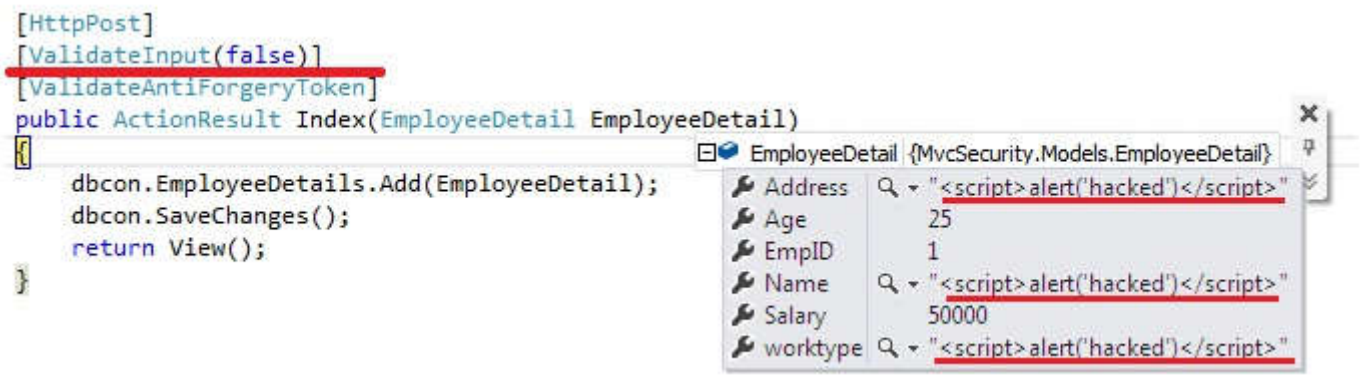
Ảnh chụp khi áp dụng ValidateInput trên phương thức HttpPost

```
[HttpPost]
[ValidateInput(false)]
[ValidateAntiForgeryToken]
public ActionResult Index(EmployeeDetail EmployeeDetail)
{
    dbcon.EmployeeDetails.Add(EmployeeDetail);
    dbcon.SaveChanges();
    return View();
}
```

Hình 3. Áp dụng **ValidateInput** trên phương thức với POST.

Sau khi đã áp dụng **ValidateInput** Attribute

Lên trên



Hình 4. Đây là đoạn code cho phép đăng script lên thông qua method POST

Giải pháp 2:

AllowHtml

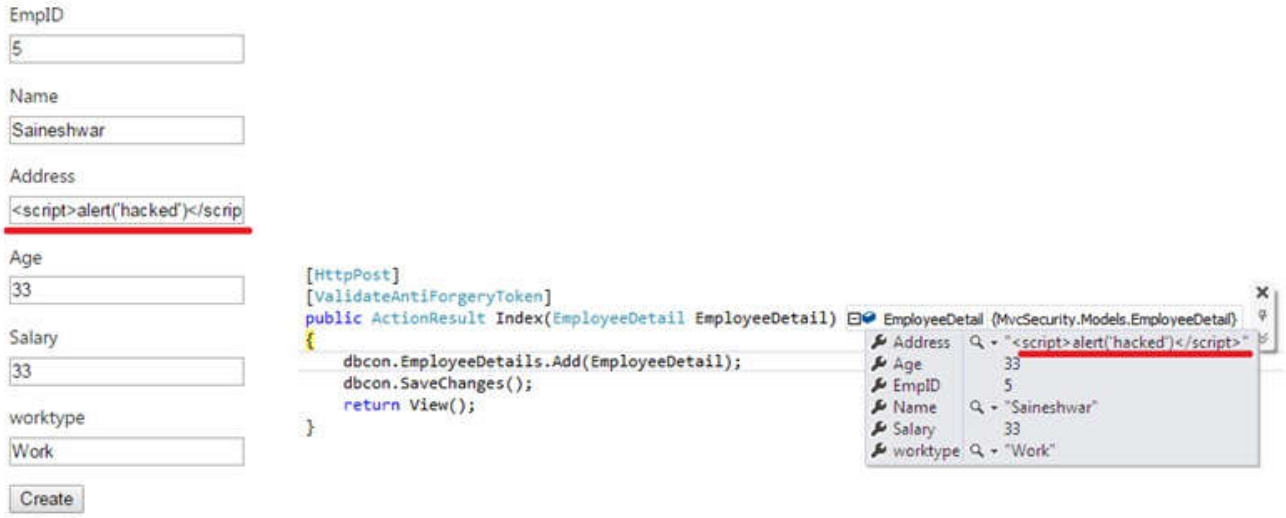
Attribute [**AllowHtml**] được áp dụng cho các thuộc tính của Model không muốn kiểm tra. Điều này cho phép đăng HTML lên hệ thống thông qua trường thuộc tính được đặt attribute này.

Đoạn code dưới đây tôi đã áp dụng **AllowHTML** trên trường Address của EmployeeDetail

Lên trên

```
public partial class EmployeeDetail
{
    public int EmpID { get; set; }
    [Required(ErrorMessage="Enter Name")]
    public string Name { get; set; }

    [StringLength(50)]
    [Required(ErrorMessage = "Enter Address")]

    [AllowHtml]
    public string Address { get; set; }

    [Required(ErrorMessage = "Enter Age")]
    public Nullable<int> Age { get; set; }

    [Required(ErrorMessage = "Enter Salary")]
    public Nullable<decimal> Salary { get; set; }

    [Required(ErrorMessage = "Enter worktype")]
    public string worktype { get; set; }
}
```

Sau khi áp dụng [AllowHtml] trên thuộc tính Address, thuộc tính Address sẽ không kiểm tra và cho phép HTML được submit vào.



Lên trên

Hình 5. Sau khi thêm [AllowHtml] vào trường Address thì có thể đăng đoạn Script này lên

Giải pháp 3:

Regular Expression

Cách thứ 3 để chống tấn công XSS là cho phép tất cả các trường đều phải validate những dữ liệu đầu vào theo một Regex. Điều này sẽ ngăn chặn việc nhập dữ liệu không mong muốn vào hệ thống.

Sử dụng Regex để kiểm tra đầu vào

```
public partial class EmployeeDetail
{
    public int EmpID { get; set; }
    [Required(ErrorMessage = "Enter Name")]
    [RegularExpression("^[A-z]+$", ErrorMessage = "Enter Valid Name")]
    public string Name { get; set; }

    [StringLength(50)]
    [RegularExpression("[a-zA-Z ]+$", ErrorMessage = "Enter Valid Address")]
    [Required(ErrorMessage = "Enter Address")]
    [AllowHtml]
    public string Address { get; set; }

    [RegularExpression("^[0-9]+$", ErrorMessage = "Enter Valid Age")]
    [Required(ErrorMessage = "Enter Age")]
    public Nullable<int> Age { get; set; }

    [RegularExpression("(((\\d+)((\\.\\d{1,2})?))$", ErrorMessage = "Enter Valid Salary")]
    [Required(ErrorMessage = "Enter Salary")]
    public Nullable<decimal> Salary { get; set; }

    [Required(ErrorMessage = "Enter Worktype")]
    [RegularExpression("[a-zA-Z ]+$", ErrorMessage = "Enter Valid Worktype")]
    public string worktype { get; set; }
}
```

Danh sách các mẫu Regex phổ biến cần thiết.

Ký tự và khoảng trắng

[a-zA-Z]+$

Chri ký tự a-z

^[A-z]+$

Lên trên

Số

^[0-9]+$

Số, chữ thường và chữ hoa

^[a-zA-Z0-9]*$

Email

[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?

Số điện thoại (tùy vùng)

^([7-9]{1})([0-9]{9})$

Ngày tháng(mm/dd/yyyy | mm-dd-yyyy | mm.dd.yyyy)

/^(0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])[- /.](19|20)\\d\\d+$/

Website URL

^http(s)?://([\\w-]+.)+[\\w-]+(/[\\w- ./?%&=])?$

Số thẻ tín dụng

Visa

^4[0-9]{12}(?:[0-9]{3})?$

MasterCard

^5[1-5][0-9]{14}$

American Express

^3[47][0-9]{13}$

Số thập phân

((\\d+)((\\.\\d{1,2})?))$

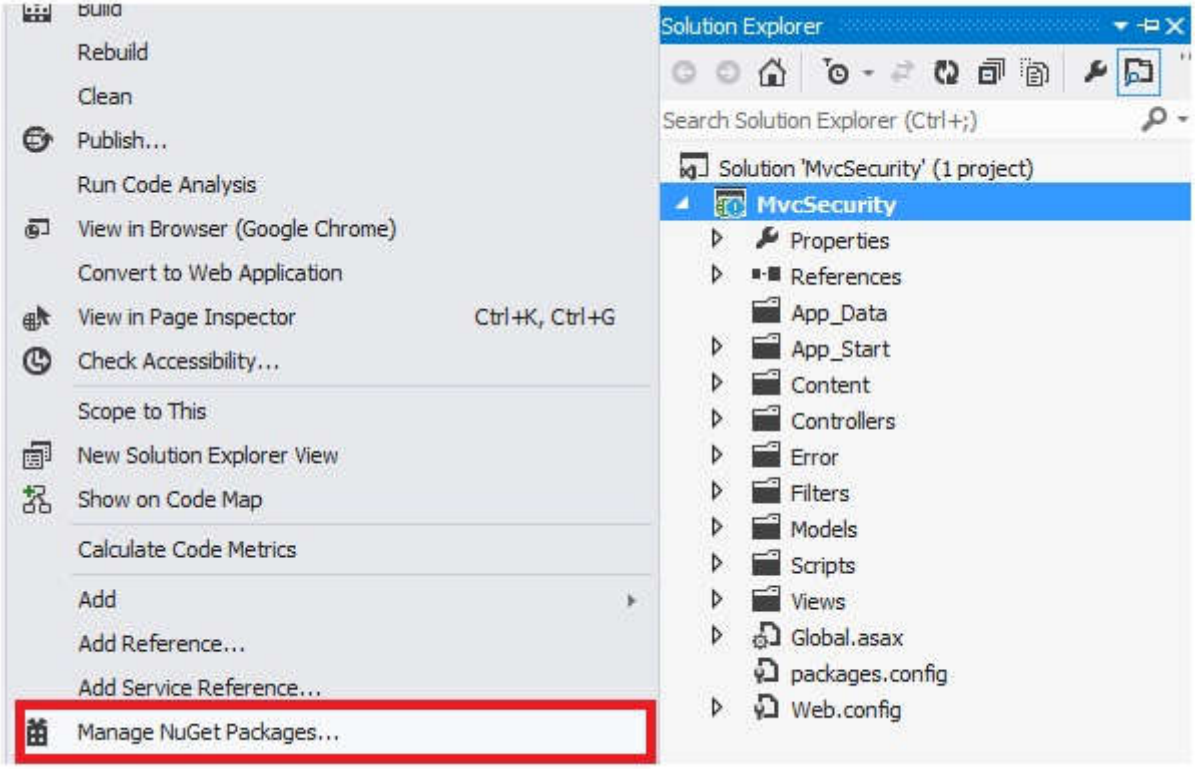
Lên trên

Giải pháp 4:-

Thư viện AntiXSS

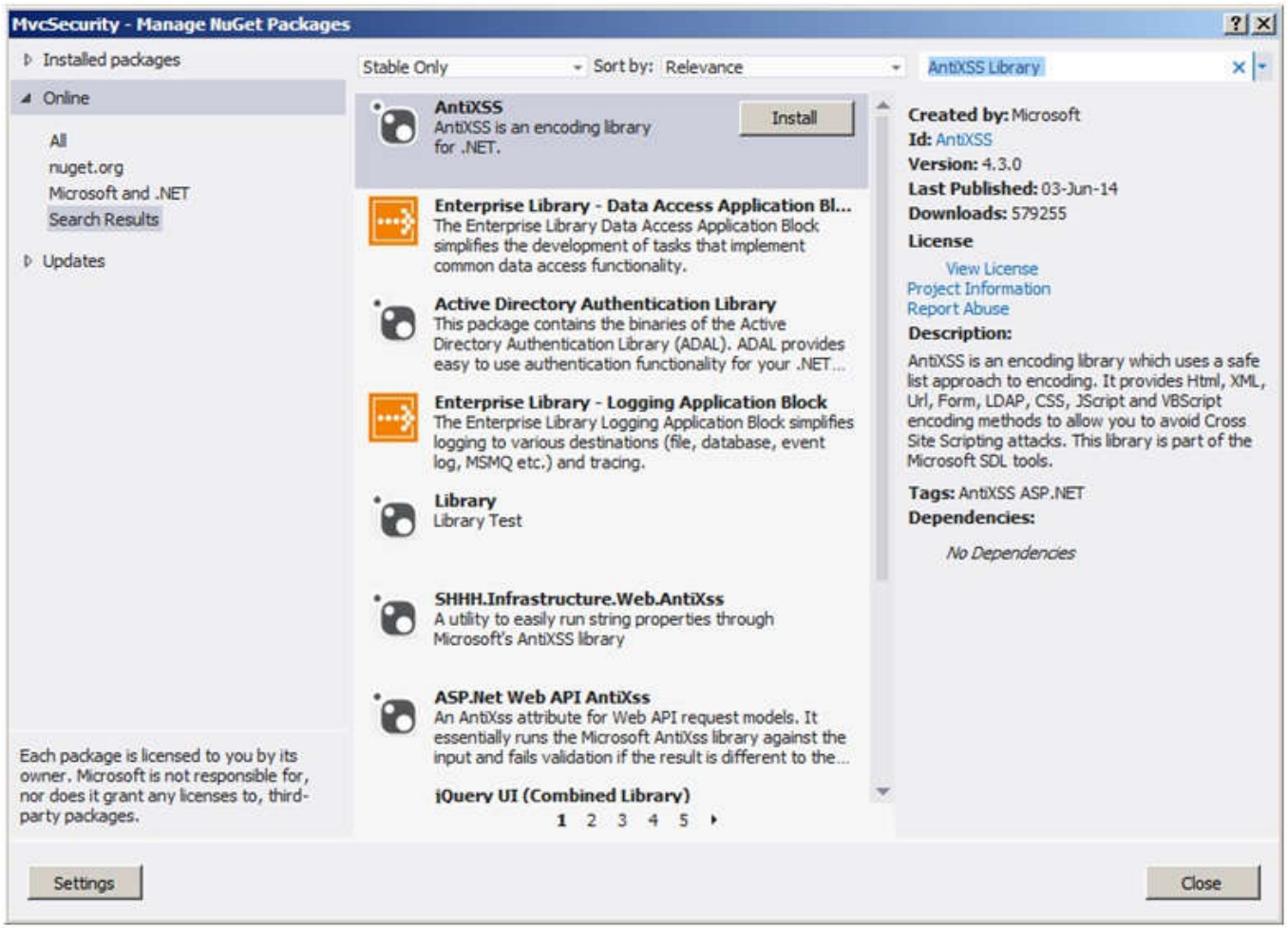
Giải pháp thứ 4 này để chống tấn công XSS bằng cách sử dụng **MicrosoftAntiXSSLibrary** sẽ giúp bảo vệ ứng dụng của bạn.

Nào giờ hãy cài đặt **MicrosoftAntiXSSLibrary** từ **Nuget** chỉ cần chuột phải vào project chọn **Manage Nuget Packages.**

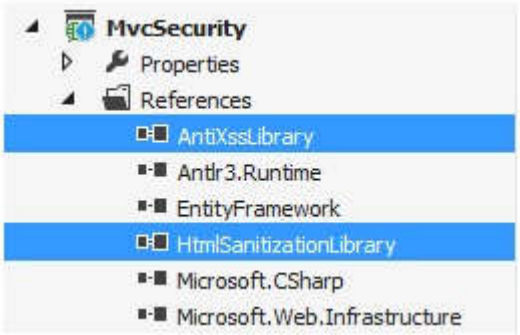


Sau khi chọn thì một cửa sổ hiển thị bạn tìm thư viện **AntiXSSLibrary** trong ô tìm kiếm và chọn **AntiXSS** sau đó click Install.

Lên trên



Sau khi cài đặt thành công



Lên trên

Sau khi cài đặt chúng ta có thể xem cách sử dụng như sau:

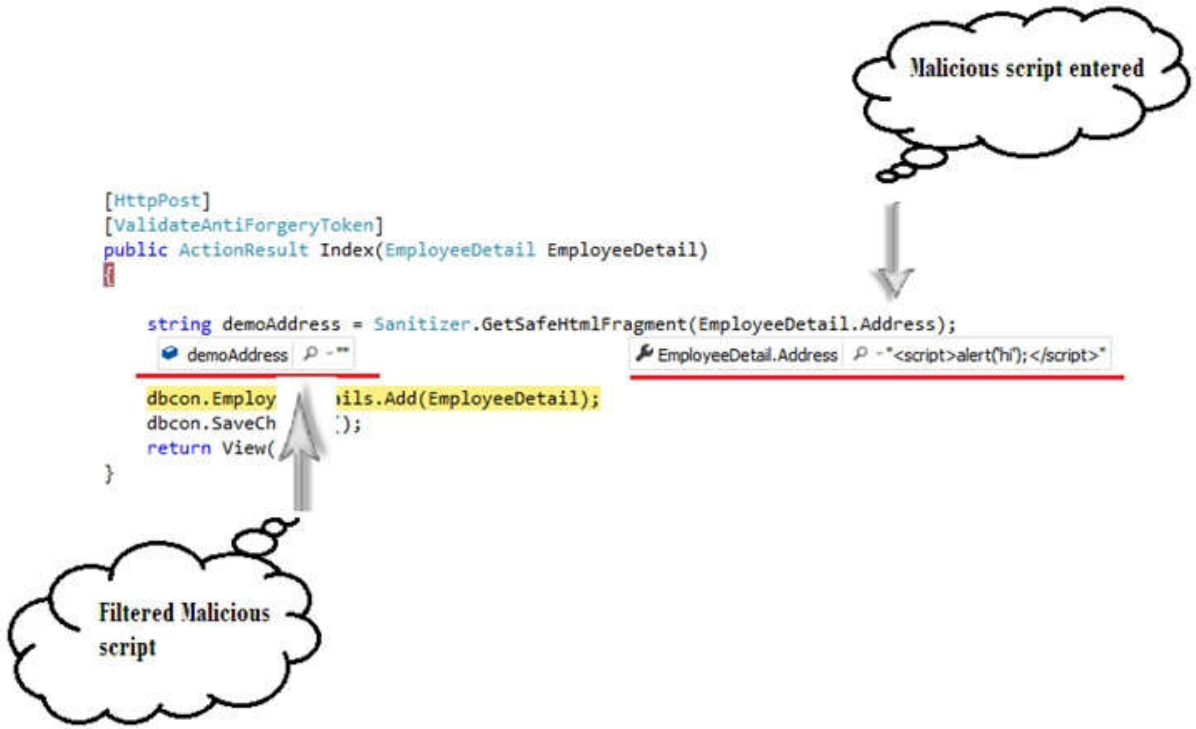
Sanitizer Class

```
public static class Sanitizer
{
    public static string GetSafeHtml(string input);
    public static void GetSafeHtml(TextReader sourceReader, Stream destinationStream);
    public static void GetSafeHtml(TextReader sourceReader, TextWriter destinationWriter);
    public static string GetSafeHtmlFragment(string input);
    public static void GetSafeHtmlFragment(TextReader sourceReader, Stream destinationStream);
    public static void GetSafeHtmlFragment(TextReader sourceReader, TextWriter destinationWriter);
}
```

Fig 27. Sanitizer Class dùng để làm sạch input

Hình dưới đây mô tả cách sử dụng của Sanitizer Class

Sanitizer là class static có thể truy cập bất cứ mọi nơi chỉ cần đưa input của trường nào vào là có thể kiểm tra được, nó sẽ kiểm tra và trả về chuỗi kết quả đã được làm sạch.



Lên trên

Chúng ta có thể sử dụng phương thức này để lọc các script có hại được đẩy vào database và hiển thị lên trình duyệt.

Thủ thuật: Sử dụng [ValidateInput(false)] hoặc [AllowHtml] trước khi sử dụng AntiXSS để tránh lỗi đưa ra: “A potentially dangerous Request.Form”

Tags

tấn công XSS (/tag/tan-cong-XSS.html)

Chia sẻ

Trích nguồn từ: (codeproject.com)

BÀI LIÊN QUAN

Làm việc với User Secrets trong ứng dụng ASP.NET Core (/bao-mat/lam-viec-voi-user-secrets-trong-ung-dung-aspnet-core-116.html)

bởi Bạch Ngọc Toàn ngày 07/03/2018

Chúng ta tìm hiểu cách làm việc với UserSecrets để quản lý các thông tin cấu hình nhạy cảm và bí mật của ứng dụng.

Xem thêm (/bao-mat/lam-viec-voi-user-secrets-trong-ung-dung-aspnet-core-116.html)

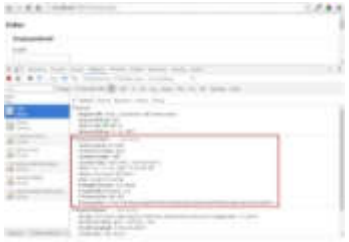
Lên trên

Tổng quan các loại thuật toán mã hoá dữ liệu (/bao-mat/tong-quan-cac-loai-thuat-toan-ma-hoa-du-lieu-106.html)

bởi Bạch Ngọc Toàn ngày 26/10/2017

Bài viết rất hay giới thiệu về tổng quan các loại thuật toán mã hoá dữ liệu dành cho những bạn nào muốn tìm hiểu về thuật toán mã hoá.

Xem thêm (/bao-mat/tong-quan-cac-loai-thuat-toan-ma-hoa-du-lieu-106.html)



Series bảo mật trong ASP.NET MVC - 5: Rỏ rỉ thông tin máy chủ (/bao-mat/series-bao-mat-trong-aspnet-mvc-5-ro-ri-thong-tin-may-chu-99.html)

bởi Bạch Ngọc Toàn ngày 08/07/2017

Các thông tin về phiên bản được sử dụng bởi một kẻ tấn công sẽ khai thác để tấn công vào hệ thống.

Xem thêm (/bao-mat/series-bao-mat-trong-aspnet-mvc-5-ro-ri-thong-tin-may-chu-99.html)

Series bảo mật trong ASP.NET MVC - 4: Upload các tệp tin có hại (/bao-mat/series-bao-mat-trong-aspnet-mvc-4-upload-cac-tep-tin-co-hai-98.html)

bởi Bạch Ngọc Toàn ngày 08/07/2017

Lên trên

Qua 3 bài trước chúng ta đã từng tìm hiểu về cách bảo vệ tất cả các trường nhập liệu. Nhưng chúng ta đã bỏ qua một trường chính là trường File Upload

Xem thêm (/bao-mat/series-bao-mat-trong-aspnet-mvc-4-upload-cac-tep-tin-co-hai-98.html)

Series bảo mật trong ASP.NET MVC - 2: Cross-Site Request Forgery (CSRF) (/bao-mat/series-bao-mat-trong-aspnet-mvc-2-cross-site-request-forgery-csrf-96.html)

bởi Bạch Ngọc Toàn ngày 05/07/2017

Cách tấn công CSRF này gọi là giả mạo request khi chúng giả mạo một request không phải từ chính website mà hacker sẽ giả lập request để gửi các thông tin lên server mà không qua hệ thống website.

Xem thêm (/bao-mat/series-bao-mat-trong-aspnet-mvc-2-cross-site-request-forgery-csrf-96.html)

Series bảo mật trong ASP.NET MVC - 1: Cấu hình Custom Error Page (/bao-mat/series-bao-mat-trong-aspnet-mvc-1-cau-hinh-custom-error-page-95.html)

bởi Bạch Ngọc Toàn ngày 05/07/2017

Trong loại tấn công này, tin tặc sẽ lấy dữ liệu từ form được gửi lên từ người dùng và thay đổi giá trị sau đó gửi dữ liệu đã được sửa lên server

Xem thêm (/bao-mat/series-bao-mat-trong-aspnet-mvc-1-cau-hinh-custom-error-page-95.html)

Lên trên

7 thủ thuật giúp bảo mật ứng dụng ASP.NET developer cần biết (/bao-mat/7-thu-thuat-giup-bao-mat-ung-dung-aspnet-developer-can-biet-47.html)

bởi Bạch Ngọc Toàn ngày 30/09/2016

Là một nhà phát triển web chuyên nghiệp, bạn phải biết được các chiêu thức để giúp ứng dụng bảo mật hơn. Trong bài viết này mình sẽ liệt kê và giải thích 7 thủ thuật bảo mật ứng dụng web.

Xem thêm (/bao-mat/7-thu-thuat-giup-bao-mat-ung-dung-aspnet-developer-can-biet-47.html)

Bài mới nhất

Tối ưu tốc độ lập trình C# - Cắt chuỗi (lap-trinh-c/toi-uu-toc-do-lap-trinh-c-cat-chuoi-200.html)

Lên trên

Tạo Dockerfile cho project ASP.NET Core, build và run Docker image (kien-thuc/tao-dockerfile-cho-project-aspnet-core-build-va-run-docker-image-198.html)

Cách cài đặt Docker trên Windows (kien-thuc/cach-cai-dat-docker-tren-windows-197.html)

Truy cập nhanh

Giới thiệu (/gioi-thieu.html)

Chính sách bảo hành (/page/chinh-sach-bao-hanh.html)

Chính sách bảo mật (/page/chinh-sach-bao-mat.html)

Chính sách vận chuyển (/page/chinh-sach-van-chuyen.html)

Hình thức thanh toán (/page/hinh-thuc-thanh-toan.html)

Đăng ký nhận tin

Hãy để TEDU gửi cho bạn những thông tin mới nhất

* Bảo mật thông tin

Gửi Đi Ngay

Lên trên

tedu.international@gmail.com (mailto:tedu.international@gmail.com)
+84 966 03 6626
Số 16/2 Phạm Thận Duật, Mai Dịch Cầu Giấy Hà Nội

Lên trên