Họ và tên: Nguyễn Khắc Ngọc

Lớp hành chính: 2010A01

Thực hành tuần 7: Security

Bài 7.1: Thực nghiệm khai thác các lỗ hổng bảo mật theo OWASP với bài thực hành 4,5,6

Mục lục

Bài thực hành MVC04

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVC04.Models;
using System.IO;
using System.Data.Entity.Infrastructure;

namespace MVC04.Controllers
{
    public class NewProductController : Controller
    {
        [HttpGet]
        public ActionResult NewProduct()
        {
            return View();
        }

        [HttpPost]
        public ActionResult NewProduct(tblProduct tblProduct)
        {
            string filename =
Path.GetFileNameWithoutExtension(tblProduct.ImageFile.FileName);
            string extension = Path.GetExtension(tblProduct.ImageFile.FileName);
            filename = filename + DateTime.Now.ToString("yymmssff") + extension;
            tblProduct.ImageURL = "~/img/" + filename;
            filename = Path.Combine(Server.MapPath("~/img/"), filename);
            tblProduct.ImageFile.SaveAs(filename);
            using(DB_EF_firtsEntities1 db = new DB_EF_firtsEntities1())
            {
                db.tblProducts.Add(tblProduct);
                db.SaveChanges();
            }
            return View();
        }

        [HttpGet]
        public ActionResult Allproduct() {
            DB_EF_firtsEntities1 db = new DB_EF_firtsEntities1();
            List<tblProduct> products = db.tblProducts.ToList();
            return View(products);
        }
    }
}
```

Đoạn mã trên là phần controller thêm sản phẩm và hiển thị danh sách tất cả sản phẩm

Khai thác các lỗ hổng bảo mật theo OWASP Top 10:

1. Injection

Khả năng cao sẽ không có lỗi vì không sử dụng câu truy vấn sql hay các thủ tục proc để lấy dữ liệu mà sử dụng Entity Framework

2. Broken Authentication and Session management

Không có lỗi vì chưa sử dụng asp.net dựa trên cookies hay session

3. Cross site scripting

Có thể bị tấn công bằng cách chèn các đoạn mã ẩn bằng javascript với các trường nhập vào khi thêm sản phẩm hiển thị trên trang web:

```
@model MVC04.Models.tblProduct

@{
    ViewBag.Title = "NewProduct";
}

<h2>Add Product</h2>

@using (Html.BeginForm("NewProduct", "NewProduct",FormMethod.Post, new { enctype =
"multipart/form-data" }))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.ProductName, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ProductName, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ProductName, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ImageURL, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
              <input type="file" name="ImageFile" required/>
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ProductPrice, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ProductPrice, new { htmlAttributes = new
{ @class = "form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.ProductPrice, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Description, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Description, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Description, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("All Product", "Allproduct")
</div>
```

Cách khắc phục:

Để xử lý các cuộc tấn công đó bằng cách mã hoá html đầu vào trước khi xử lý nó

```
protected void btnSubmit_Click(object sender, EventArgs e)
    {
        Response.Write(Server.HtmlEncode(txtMessage.Text));
    }
```

4. Insecure Direct Object References

Không bị vì cơ chế navigate của asp.net mvc đã được định nghĩa khá rõ ràng

5. Security Misconfiguration

Không có lỗi vì đã cấu hình bảo mật đúng như các connectionString trong webconfig hay không sửa đổi gì trong file web.config

6. Sensitive Data Exposure

Không có lỗi gì không hề sử dụng gì đến các thông tin nhạy cảm, thông tin KYC hay các thông tin về vấn đề thanh toán

7. Missing Function Level Access Controll

Với bài thực hành MVC04 mới chỉ là 1 ứng dụng nhỏ làm các chức năng thêm sản phẩm và hiển thị danh sách sản phẩm, chưa hề dùng đến các chức năng phân quyền nên sẽ không có lỗi ở phần này

8. Cross-Site Request Forgery

Có thể bị tấn công với các đoạn mã javascript tạo các cookies với yêu cầu Post để thêm sản phẩm.

Cách khắc phục có thể là:

Sử dụng AntiForgery Token trong ASP.NET MVC View/Controller

```
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @*Removed for brevity *@
}
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(Employee emp)
{
```

9. Using Components with Known Vulnerabilities

Không có lỗi vì toàn bộ mã nguồn đều tự viết và kiểm soát, không sử dụng các thành phần mã nguồn mở bên ngoài nên sẽ không bị tấn công ở phần này

10. Under Protected APIs

Không có lỗi vì không sử dụng API bên thứ 3 trong mã nguồn

Bài thực hành MVC05

Bài thực hành MVC05 bổ sung thêm phần xoá 1 sản phẩm trên 1 danh sách sản phẩm sử dụng ajax vì vậy phần thực nghiệm cho bài này sẽ chỉ là bổ sung thêm nếu có. Trường hợp của bài này là không có

Bài thực hành MVC06

Bài thực hành MVC06 cũng tiếp tục bổ sung thêm chức năng lọc sản phẩm và thêm sản phẩm vào giỏ hàng với ajax

Bổ sung thực nghiệm khai thác lỗi cho bài này:

8. Cross-Site Request Forgery

Có thể bị tấn công với các đoạn mã javascript tạo các cookies với yêu cầu POST để thêm sản phẩm vào giỏ hàng

```
document.addEventListener("DOMContentLoaded", function () {
    document.querySelector(".add-to-cart-btn").addEventListener("click", function () {
        var hanghoaID = this.getAttribute("data-hanghoa-id")
        var xhr = new XMLHttpRequest()
        xhr.open("POST", "/HangHoa/addtoCarts")
        xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
        var data = JSON.stringify({ hanghoaID: hanghoaID })
        xhr.onload = function () {
            if (xhr.status === 200) {
                alert("Thêm thành công")
            }
            else alert("Đã có lỗi xảy ra")
        }
        xhr.send(data)
    })
```

})Cách khắc phục có thể là:

Sử dụng AntiForgery Token trong ASP.NET MVC View/Controller

```
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @*Removed for brevity *@
}
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(Employee emp)
{
```

Tài liệu tham khảo:

1. https://owasp.org/www-pdf-archive/OTD_2014_-_OWASPTop10forMVC.pdf
2. http://www.codedigest.com/quick-start/15/what-is-owasp-what-are-owasp-top-10-security-risks