**BÁO CÁO BÀI THỰC HÀNH SỐ 9**

**KIỂM THỬ LỖ HỔNG XSS và CSRF**

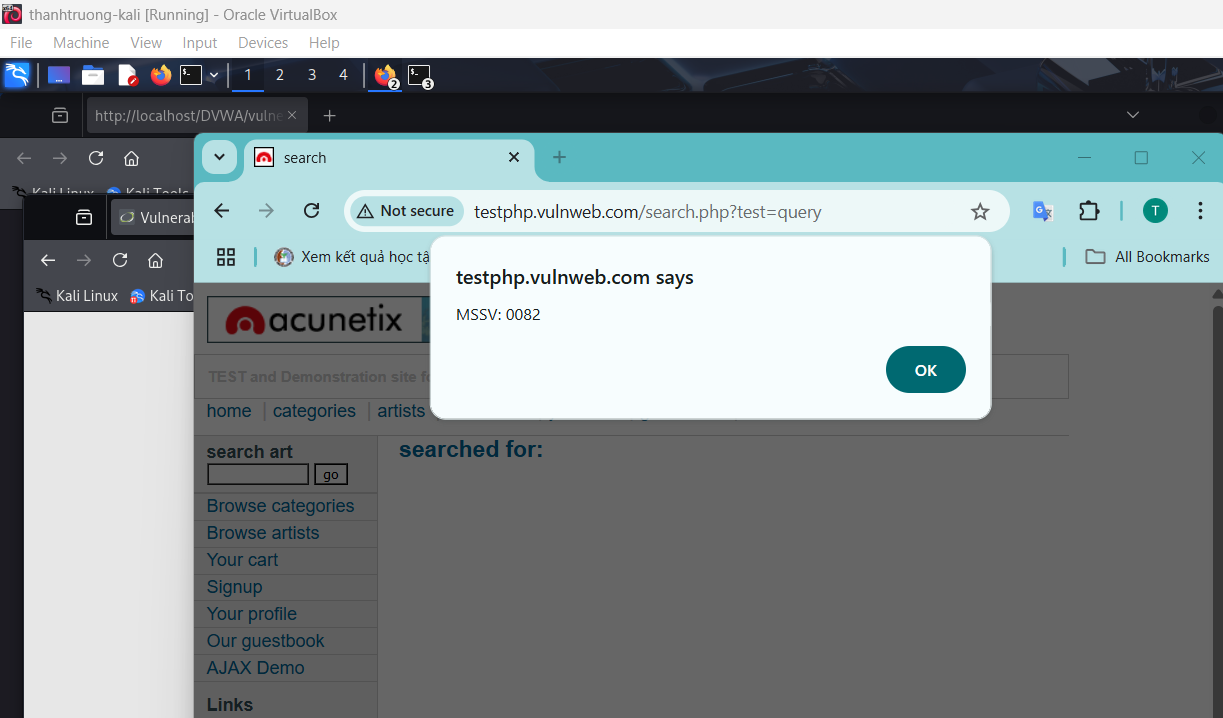
***Họ và tên: Nguyễn Thanh Trường***

***MSSV: 1050080082***

**Câu 1 Thực hiện trên** [http://testphp.vulnweb.com](http://testphp.vulnweb.com/)

Thực hiện kiểm thử lỗ hổng XSS(Reflected) để thực thi đoạn mã hiển thị thông báo có chứa 4 số cuối MSSV.

- Bước 1: Nhập vào ô nhập liệu giá trị <script>alert(“MSSV: 0082")</script> chúng ta được kết quả như sau:



Hãy giải thích:

**-** Xác định các tham số đầu vào:

Ctrl + U ta sẽ thấy được có một form



Tham số đầu vào là: searchFor, được truyền qua POST(nằm trên URL sau khi gửi form).

- Xác định giá trị đầu vào xuất hiện trong trang kết quả: **0.25 điểm**

Sau khi bạn nhập một giá trị và nhấn "Submit", trang sẽ phản hồi lại: Hello + giá trị bạn nhập

Điều này có nghĩa là giá trị của name được chèn trực tiếp vào HTML, cho thấy đây là nơi dễ bị khai thác XSS nếu không được xử lý đúng cách.

**-** Các bước kiểm thử và phán đoán: **0-1 điểm** theo các bước kiểm thử và logic trong phán đoán

* Sau khi nhập đoạn mã <script>alert(“MSSV: 0082”)</script> vào ô input
* Nhấn “submit”
* Quan sát kết quả ta thấy hộp thoại hiển thị nội dung: MSSV: 0082 => bị Xss thành công

**Phán đoán:**

* Nếu mức độ bảo mật của đang ở Low: chắc chắn đoạn script sẽ được thực thi.
* Nếu đang ở Medium/High/Impossible: sẽ thực hiện kiểm tra hoặc mã hóa đầu vào, chặn được XSS.

- Giá trị kiểm thử cho thấy lỗ hổng và giải thích: **0.5 điểm**

* Giá trị kiểm thử là : <script>alert(“MSSV: 0082”)</script>

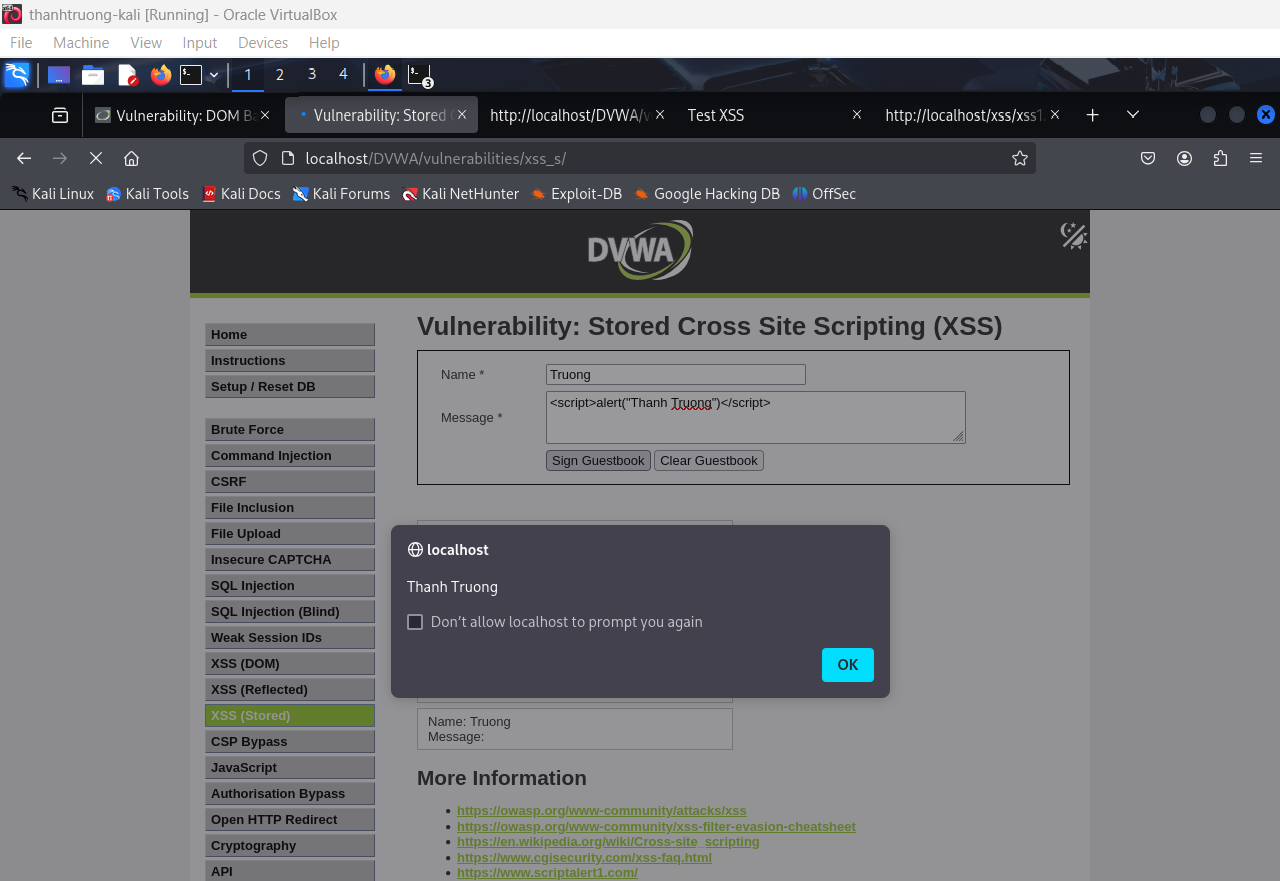
Giải thích:

* Đây là một đoạn mã JavaScript hợp lệ, dùng để kiểm tra xem giá trị đầu vào có được lọc/escape hay không trước khi phản hồi lại trình duyệt.
* Dữ liệu đầu vào **được chèn trực tiếp vào HTML** mà **không qua kiểm tra hoặc mã hóa**.
* Trình duyệt thực thi đoạn JavaScript nằm trong đầu vào của người dùng, chứng tỏ ứng dụng bị **lỗ hổng XSS dạng phản chiếu** (Reflected XSS).

**Câu 2**

Thực hiện kiểm thử lỗ hổng XSS (Stored) để thực thi đoạn mã hiển thị thông báo có chứa tên của sinh viên. Điểm cho mỗi tham số đầu vào đã kiểm thử được:

Bước 1: Nhập vào ô nhập liệu giá trị <script>alert(“Thanh Truong”)</script> chúng ta được kết quả như sau:



Hãy giải thích:

**-** Xác định các tham số đầu vào:

Ctrl + U ta sẽ thấy được có một form



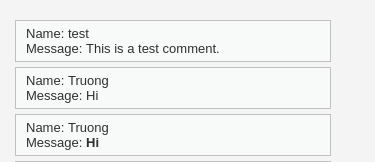
Tham số đầu vào là: txtName và txtMessage , được truyền qua POST (nằm trên URL sau khi gửi form).

- Xác định giá trị đầu vào xuất hiện trong trang kết quả: **0.25 điểm**

Sau khi bạn nhập một giá trị và nhấn "Submit", trang sẽ phản hồi lại:

**Name + giá trị bạn nhập**

**Message + giá trị bạn nhập**

****

Điều này có nghĩa là giá trị của name được chèn trực tiếp vào HTML, cho thấy đây là nơi dễ bị khai thác XSS nếu không được xử lý đúng cách.

**-** Các bước kiểm thử và phán đoán: **0-1 điểm** theo các bước kiểm thử và logic trong phán đoán

* Sau khi nhập đoạn mã <script>alert(“Thanh Truong”)</script> vào ô message
* Nhấn “submit” Quan sát kết quả ta thấy hộp thoại hiển thị nội dung: Thanh Truong
* Những lần truy cập tiếp theo vào trang này sẽ luôn luôn hiển thị hộp thoại Thanh Truong => DVWA đã bị XSS stored thành công

**Phán đoán:**

* Vì đoạn mã JavaScript được lưu trữ vĩnh viễn trên server và tự động thực thi khi người dùng (bao gồm admin hoặc người dùng khác) truy cập trang, điều này chứng tỏ ứng dụng **dính lỗ hổng Stored XSS**. Đây là loại XSS nguy hiểm hơn Reflected XSS vì nó không cần nạn nhân nhấn vào liên kết – mã độc sẽ tự chạy mỗi lần truy cập.

- Giá trị kiểm thử cho thấy lỗ hổng và giải thích: **0.5 điểm**

* Giá trị kiểm thử là :<script>alert("Thanh Truong")</script>

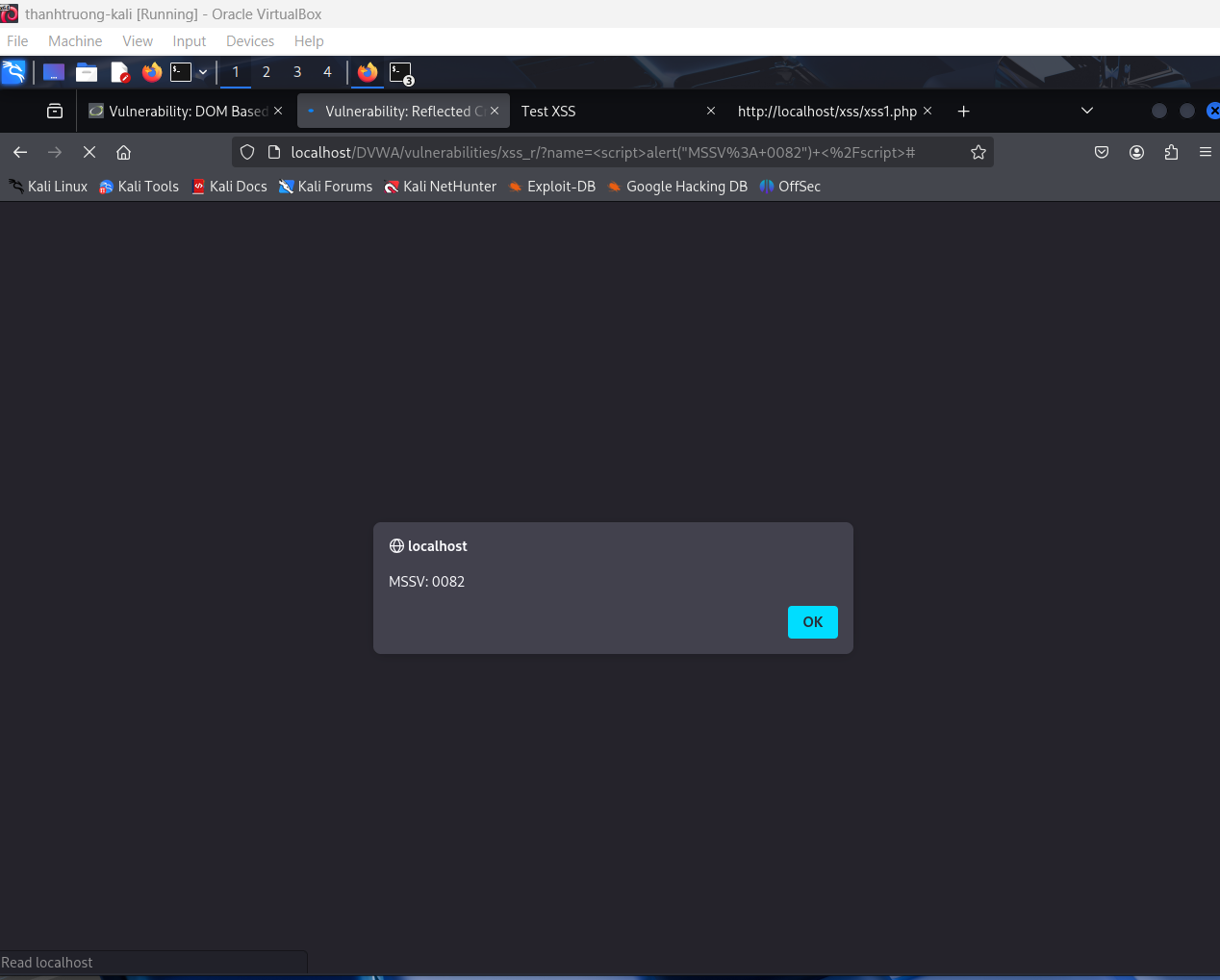
Giải thích:

* Đoạn mã kiểm thử này là một đoạn JavaScript đơn giản dùng để bật hộp thoại thông báo.
* Khi nhập và gửi đoạn này trong phần Stored XSS:
  + Nếu hộp thoại xuất hiện mỗi lần truy cập lại trang, thì điều đó chứng minh rằng:
    - Ứng dụng đã lưu trữ nội dung đầu vào người dùng mà không lọc/mã hóa dữ liệu.
    - Nội dung này được chèn trực tiếp vào mã HTML của trang mà không escape các thẻ HTML/JS.
    - Do đó, trình duyệt sẽ thực thi đoạn mã, gây ra lỗ hổng XSS dạng lưu trữ (Stored XSS).

**Câu 3**

Thực hiện kiểm thử lỗ hổng XSS để thực thi đoạn mã hiển thị thông báo có chứa 4 số cuối MSSV. Hãy giải thích:

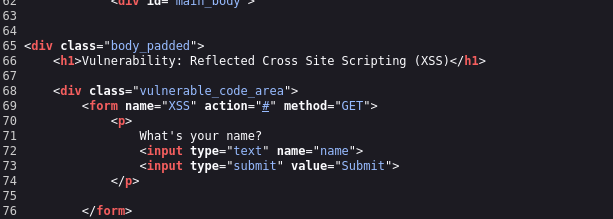
Bước 1: Nhập vào ô nhập liệu giá trị <script>alert(“MSSV: 0082")</script> chúng ta được kết quả như sau:



Hãy giải thích:

**-** Xác định các tham số đầu vào:

Ctrl + U ta sẽ thấy được có một form



Tham số đầu vào là: name , được truyền qua GET (nằm trên URL sau khi gửi form).

- Xác định giá trị đầu vào xuất hiện trong trang kết quả: **0.25 điểm**

Sau khi bạn nhập một giá trị và nhấn "Submit", trang sẽ phản hồi lại: Hello + giá trị bạn nhập

Điều này có nghĩa là giá trị của name được chèn trực tiếp vào HTML, cho thấy đây là nơi dễ bị khai thác XSS nếu không được xử lý đúng cách.

**-** Các bước kiểm thử và phán đoán: **0-1 điểm** theo các bước kiểm thử và logic trong phán đoán

* Sau khi nhập đoạn mã <script>alert(“MSSV: 0082”)</script> vào ô input
* Nhấn “submit” thì url sẽ chuyển thành http://localhost/dvwa/vulnerabilities/xss/?name=<script>alert(“MSSV%3A+0082”)</script>
* Quan sát kết quả ta thấy hộp thoại hiển thị nội dung: MSSV: 0082 =>DVWA bị Xss thành công

**Phán đoán:**

* Nếu mức độ bảo mật của DVWA đang ở Low: chắc chắn đoạn script sẽ được thực thi.
* Nếu đang ở Medium/High/Impossible: DVWA sẽ thực hiện kiểm tra hoặc mã hóa đầu vào, chặn được XSS.

- Giá trị kiểm thử cho thấy lỗ hổng và giải thích: **0.5 điểm**

* Giá trị kiểm thử là : <script>alert(“MSSV: 0082”)</script>

Giải thích:

* Đây là một đoạn mã JavaScript hợp lệ, dùng để kiểm tra xem giá trị đầu vào có được lọc/escape hay không trước khi phản hồi lại trình duyệt.
* Dữ liệu đầu vào **được chèn trực tiếp vào HTML** mà **không qua kiểm tra hoặc mã hóa**.
* Trình duyệt thực thi đoạn JavaScript nằm trong đầu vào của người dùng, chứng tỏ ứng dụng bị **lỗ hổng XSS dạng phản chiếu** (Reflected XSS).

**Phần 2: Kiểm thử CSRF (2 điểm)**

Trình bày các bước thực hiện và kết quả kiểm thử lỗ hổng CSRF trên website chọn 1 trong các trang sau để thử nghiệm

1. [http://demo.testfire.net](http://demo.testfire.net/)
2. [http://php.testsparker.com](http://php.testsparker.com/)
3. [http://testasp.vulnweb.com](http://testasp.vulnweb.com/)
4. [http://testasp.vulnweb.com](http://testasp.vulnweb.com/)
5. [http://testphp.vulnweb.com](http://testphp.vulnweb.com/)
6. [http://www.webscantest.com](http://www.webscantest.com/)
7. [http://testhtml5.vulnweb.com](http://testhtml5.vulnweb.com/)
8. [http://aspnet.testsparker.com](http://aspnet.testsparker.com/)
9. [http://zero.webappsecurity.com](http://zero.webappsecurity.com/)