**BÁO CÁO BÀI THỰC HÀNH SỐ 9**

**KIỂM THỬ LỖ HỔNG XSS và CSRF**

***Họ và tên:* Phan Tân Phước**

**MSSV: 1050080070**

**Lớp: 10 – ĐH – CNPM 1**

**Link:** [**https://youtu.be/90iwL91OJ0c**](https://youtu.be/90iwL91OJ0c)

***Yêu cầu kết quả thực hành:***

Báo cáo kết quả thực hành có tên theo định dạng HoTenSV\_MSSV\_Lab06.docx

Trình bày quá trình kiểm thử theo mẫu sau

**Phần 1: Kiểm thử XSS**

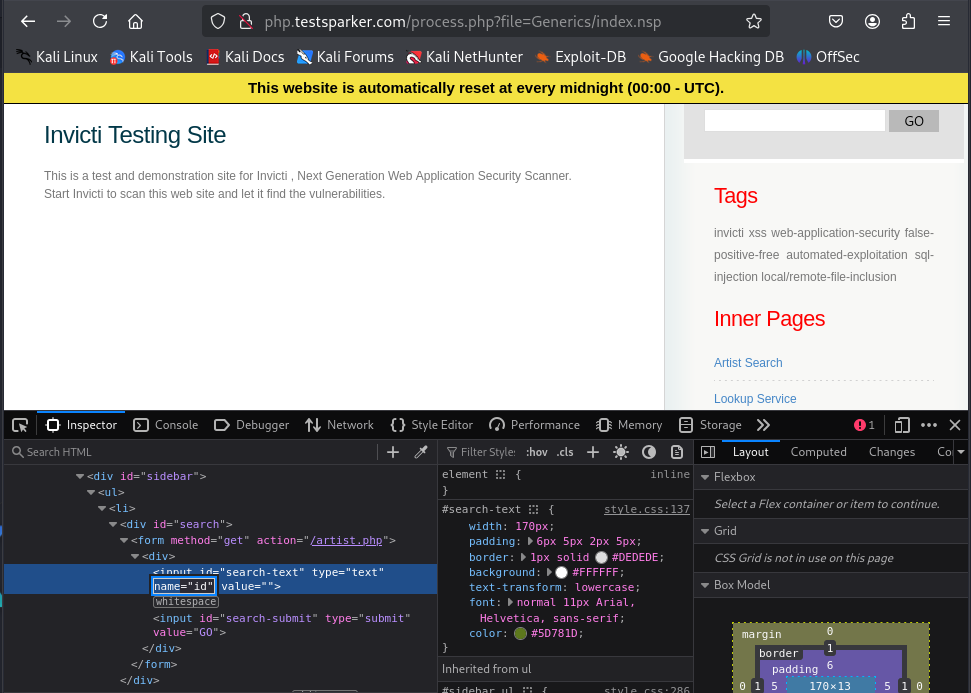
Thực hiện kiểm thử lỗ hổng XSS trên website: chọn 1 trong các trang sau để thử nghiệm

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/)

**Câu 1**

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

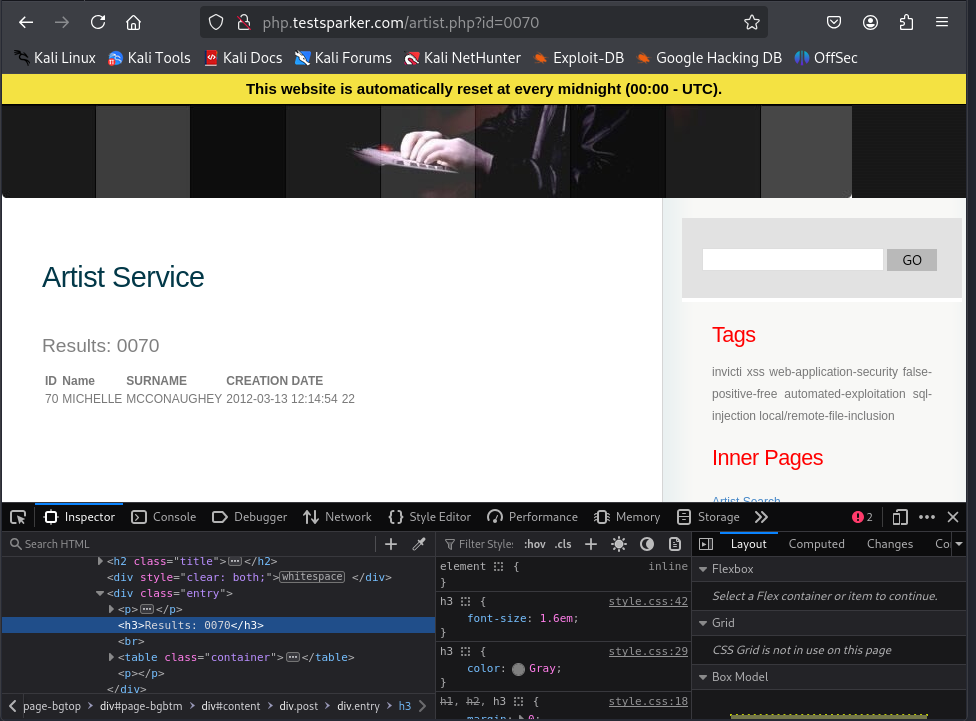
**1.1.** Xác định các tham số đầu vào: **0.25 điểm**



- Phân tích URL và xác định tham số truyền vào là id.

- Đây là tham số dạng GET → có khả năng được server xử lý và xuất ra lại giao diện

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

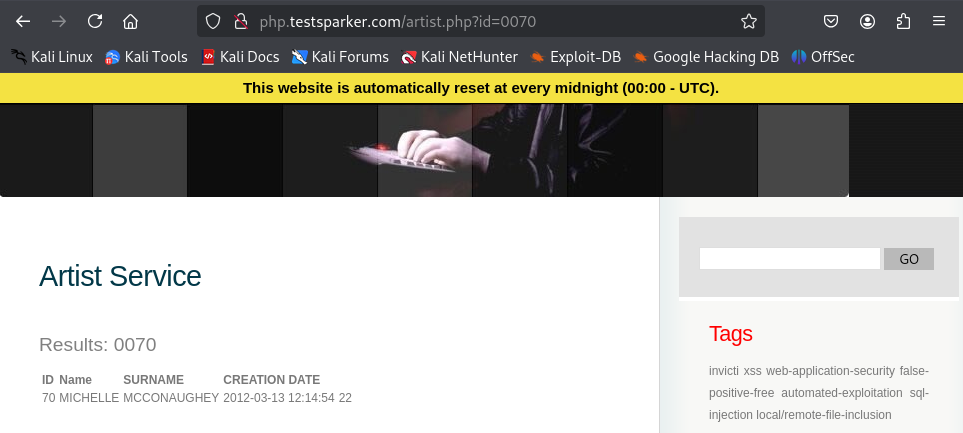
****

- Điều này cho thấy giá trị của tham số id được phản chiếu (reflected) trên giao diện HTML, có thể là bên trong thẻ <h3>

- Dẫn đến khả năng tồn tại lỗ hổng XSS phản chiếu (Reflected XSS).

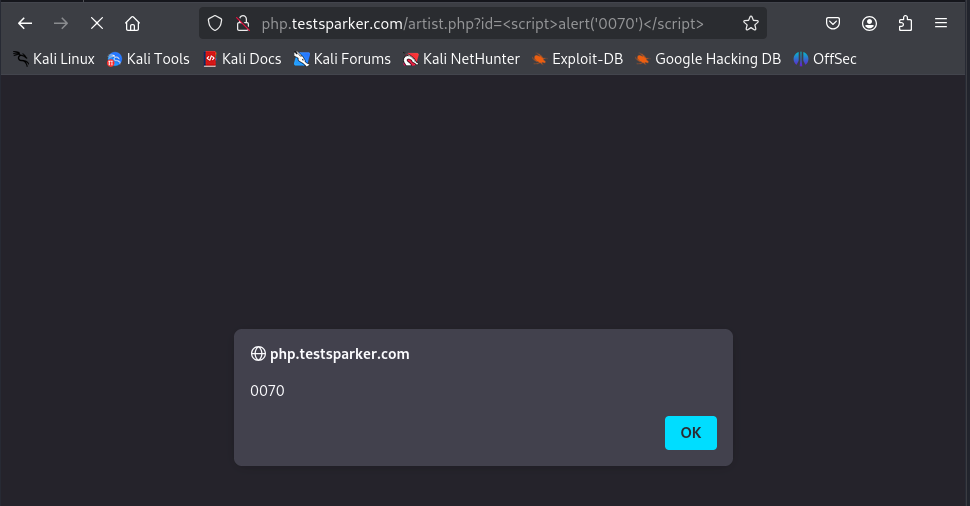
1.3.Các bước kiểm thử và phán đoán: **0-1 điểm** theo các bước kiểm thử và logic trong phán đoán

Bước 1: Kiểm tra đầu vào có được phản chiếu lại không. Truy cập: ?id=0070 → Thấy 0070 xuất hiện trên giao diện.



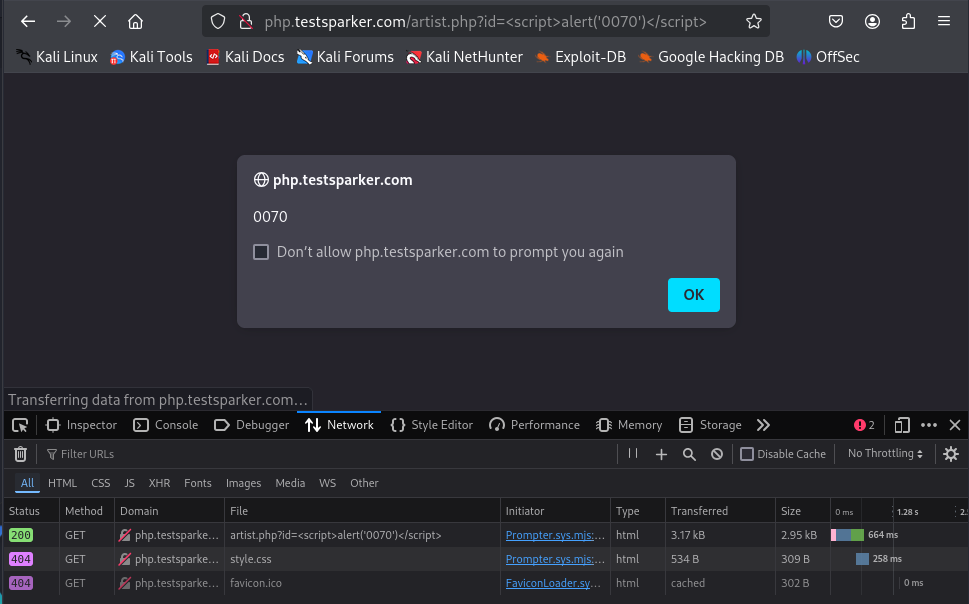
Bước 2: Thử chèn đoạn mã XSS. Truy cập URL: [http://php.testsparker.com/artist.php?id=<script>alert('0070')</script](http://php.testsparker.com/artist.php?id=%3cscript%3ealert('0070')%3c/script)>

Bước 3: Kết quả kiểm thử. Trình duyệt hiện ra hộp thoại popup với nội dung 0070. Điều này chứng minh đoạn mã JavaScript đã được thực thi thành công.



→ Dữ liệu đầu vào từ id không được xử lý an toàn (không escape) trước khi hiển thị lên trang → tồn tại lỗ hổng XSS phản chiếu.

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

****

- Do giá trị từ tham số id được nhúng trực tiếp vào mã HTML mà không mã hóa đặc biệt (<, >, ', ",...) nên trình duyệt hiểu đó là mã JavaScript thực thi.

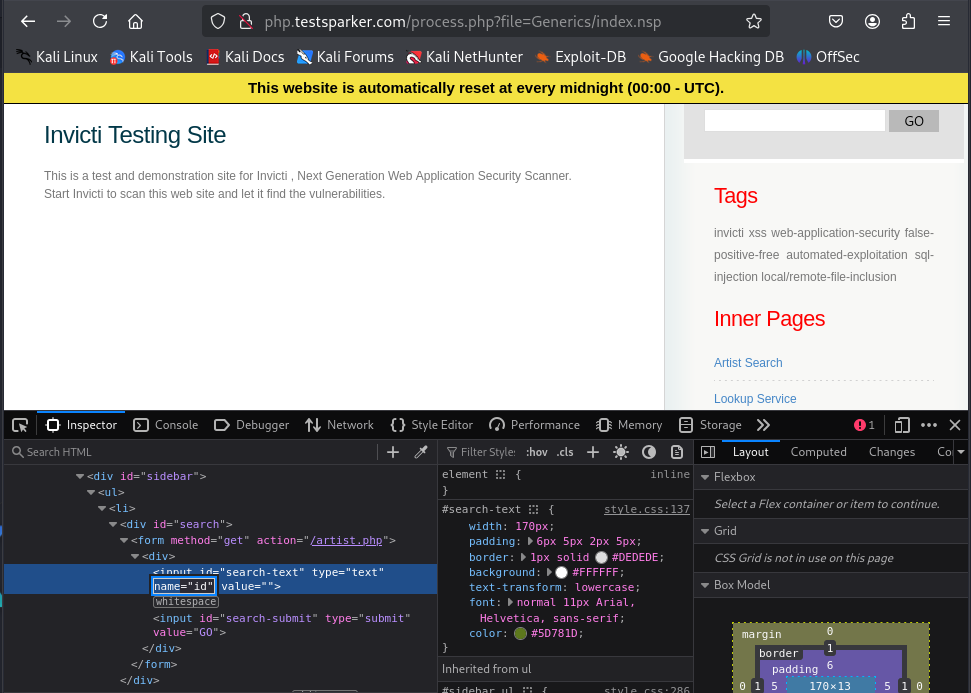
- Hacker có thể lợi dụng để:

* Đánh cắp cookie người dùng.
* Chèn mã độc redirect.
* Gài keylogger, phishing,…

**Câu 2**

Thực hiện kiểm thử lỗ hổng XSS để thực thi đoạn mã hiển thị thông báo có chứa tên của sinh viên. Tại: http://php.testsparker.com .Điểm cho mỗi tham số đầu vào đã kiểm thử được:

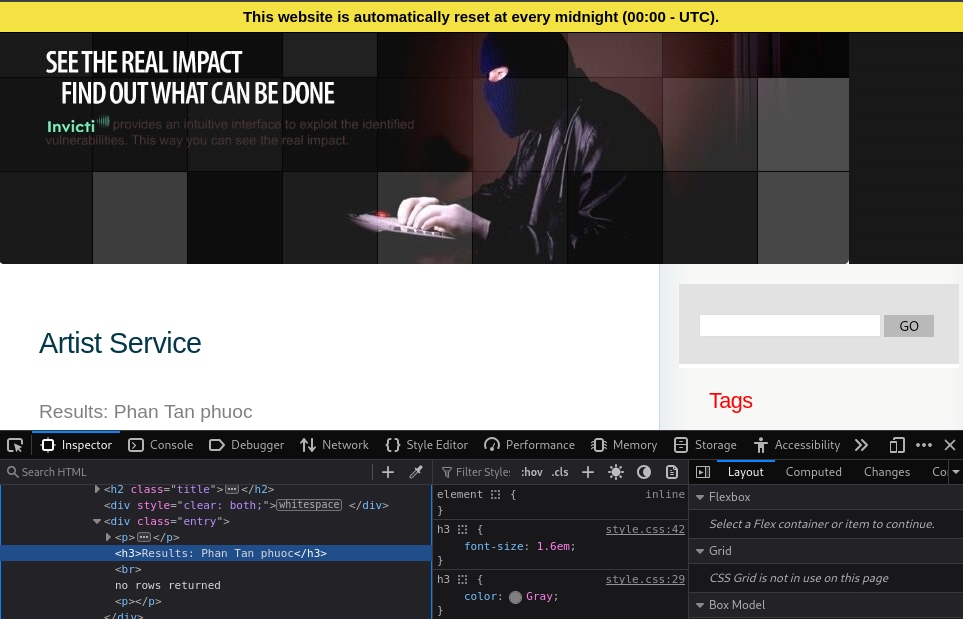
**1.1.** Xác định các tham số đầu vào: **0.25 điểm**



- Phân tích URL và xác định tham số truyền vào là id.

- Đây là tham số dạng GET → có khả năng được server xử lý và xuất ra lại giao diện

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

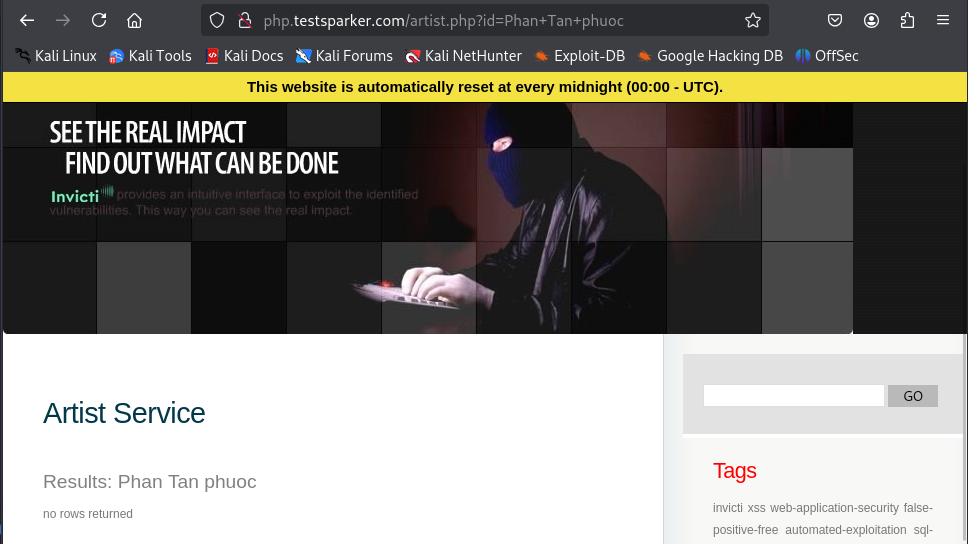
****

- Điều này cho thấy giá trị của tham số id được phản chiếu (reflected) trên giao diện HTML, có thể là bên trong thẻ <h3>

- Dẫn đến khả năng tồn tại lỗ hổng XSS phản chiếu (Reflected XSS).

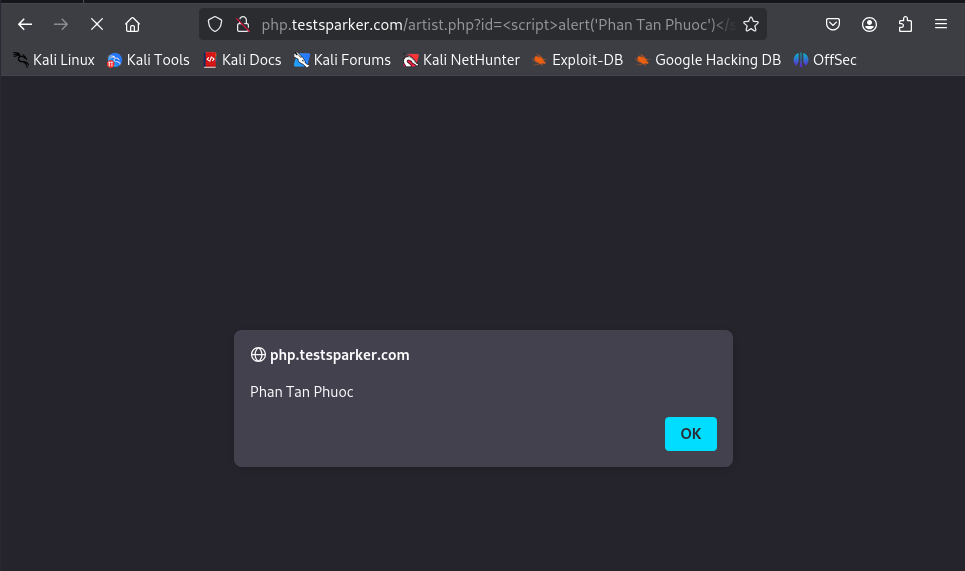
1.3.Các bước kiểm thử và phán đoán: **0-1 điểm** theo các bước kiểm thử và logic trong phán đoán

Bước 1: Kiểm tra đầu vào có được phản chiếu lại không. Truy cập: ?id=0070 → Thấy 0070 xuất hiện trên giao diện.



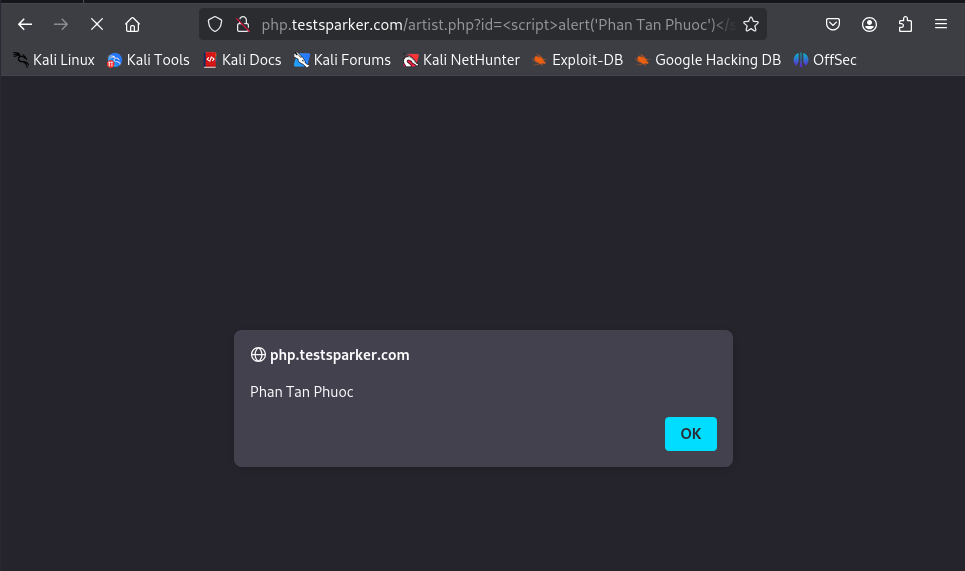
Bước 2: Thử chèn đoạn mã XSS. Truy cập URL: [http://php.testsparker.com/artist.php?id=<script>alert('Phan Tan Phuoc')</script](http://php.testsparker.com/artist.php?id=%3cscript%3ealert('Phan%20Tan%20Phuoc')%3c/script)>

Bước 3: Kết quả kiểm thử. Trình duyệt hiện ra hộp thoại popup với nội dung “Phan Tan Phuoc”. Điều này chứng minh đoạn mã JavaScript đã được thực thi thành công.



→ Dữ liệu đầu vào từ id không được xử lý an toàn (không escape) trước khi hiển thị lên trang → tồn tại lỗ hổng XSS phản chiếu.

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

**

- Do giá trị từ tham số id được nhúng trực tiếp vào mã HTML mà không mã hóa đặc biệt (<, >, ', ",...) nên trình duyệt hiểu đó là mã JavaScript thực thi.

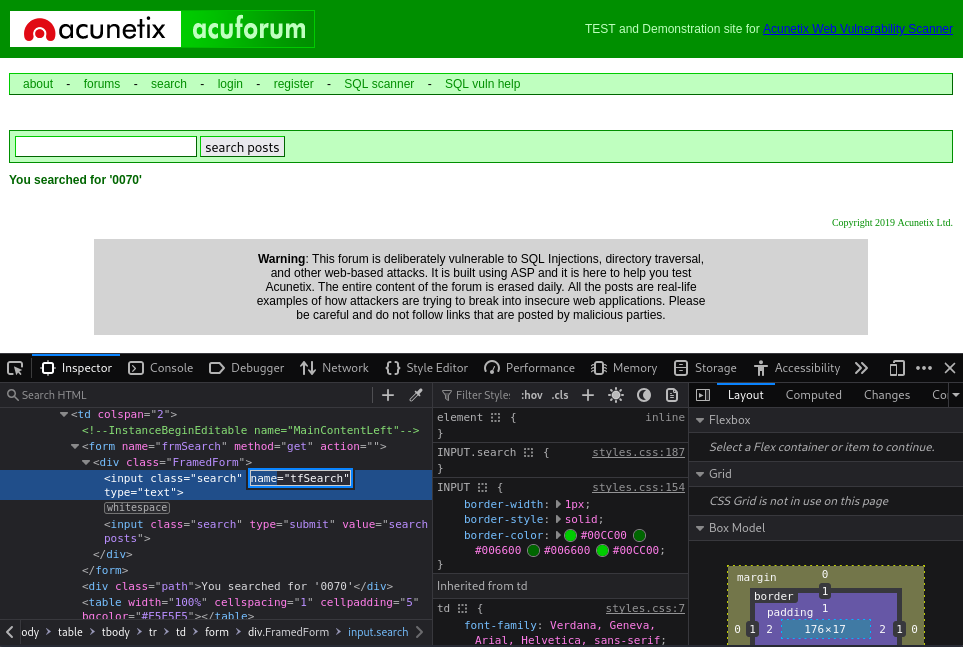
- Hacker có thể lợi dụng để:

* Đánh cắp cookie người dùng.
* Chèn mã độc redirect.
* Gài keylogger, phishing,…

**Câu 3:**

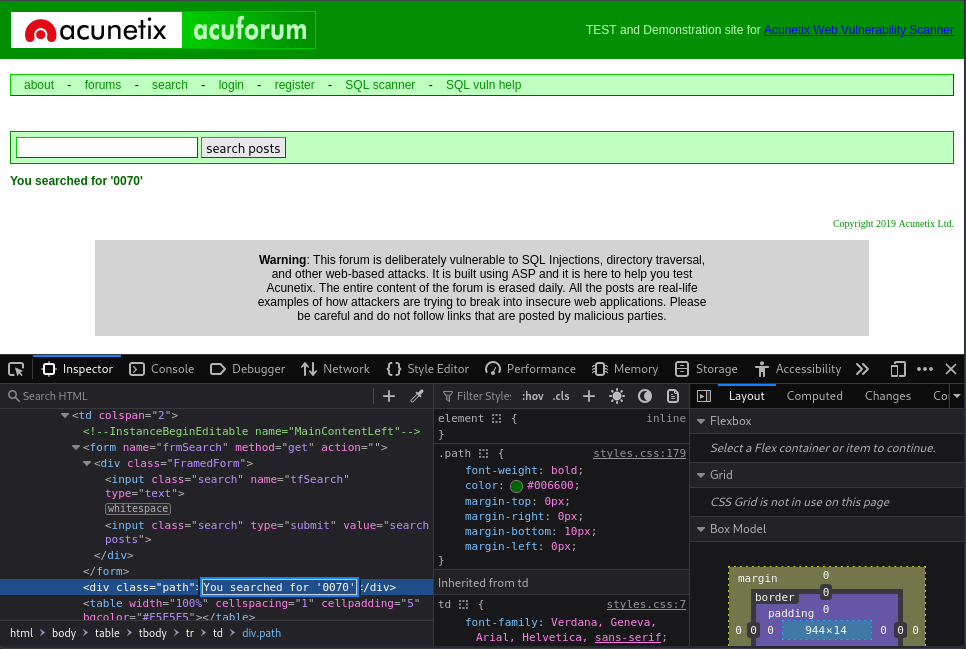
Thực hiện kiểm thử lỗ hổng XSS để thực thi đoạn mã hiển thị thông báo có chứa 4 số cuối MSSV. <http://testasp.vulnweb.com>. Hãy giải thích:

**3.1.** Xác định các tham số đầu vào: **0.25 điểm**



- Tham số đầu vào của input có tên là tfSearch.

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

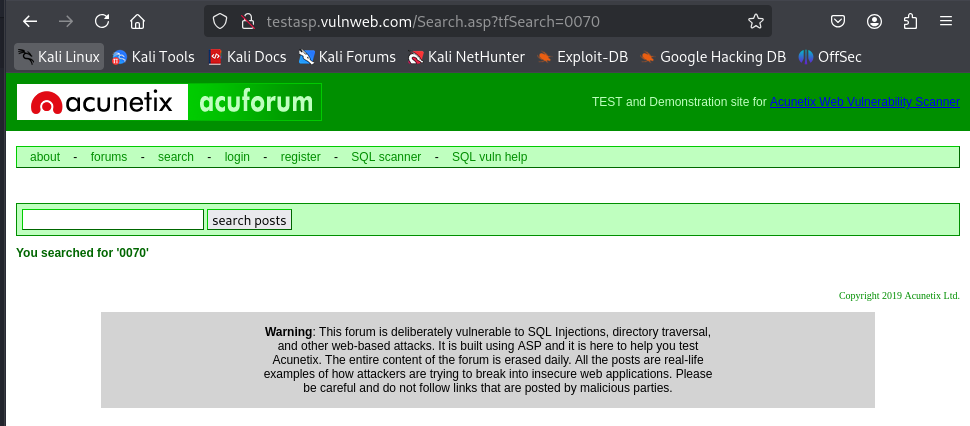


- Điều này cho thấy giá trị của tham số tfSearch được phản chiếu (reflected) trên giao diện HTML, có thể là bên trong thẻ <div>

- Dẫn đến khả năng tồn tại lỗ hổng XSS phản chiếu (Reflected XSS).

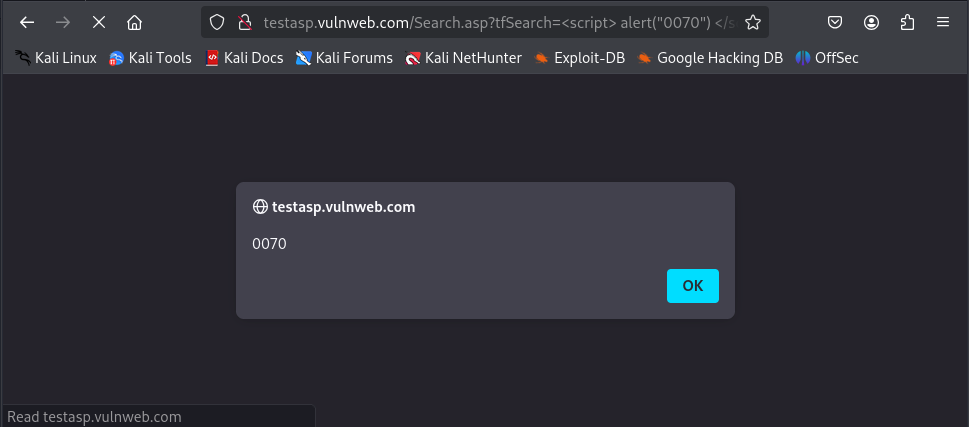
**3.3.** Các bước kiểm thử và phán đoán: **0-1 điểm** theo các bước kiểm thử và logic trong phán đoán

Bước 1: Kiểm tra đầu vào có được phản chiếu lại không. Truy cập: ?tfSearch=0070 → Thấy 0070 xuất hiện trên giao diện.



Bước 2: Thử chèn đoạn mã XSS. Truy cập URL: testasp.vulnweb.com/Search.asp?tfSearch=<script> alert("0070") </script>

Bước 3: Kết quả kiểm thử. Trình duyệt hiện ra hộp thoại popup với nội dung “Phan Tan Phuoc”. Điều này chứng minh đoạn mã JavaScript đã được thực thi thành công.



→ Dữ liệu đầu vào từ id không được xử lý an toàn (không escape) trước khi hiển thị lên trang → tồn tại lỗ hổng XSS phản chiếu.

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

- Do giá trị từ tham số id được nhúng trực tiếp vào mã HTML mà không mã hóa đặc biệt (<, >, ', ",...) nên trình duyệt hiểu đó là mã JavaScript thực thi.

**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/)