BÁO CÁO BÀI TẬP LÝ THUYẾT

**Môn học: Bảo mật web và ứng dụng**

**Kỳ báo cáo: Bài tập 1**

**Tên chủ đề: XSS-CSRF-CORS**

*GVHD: Nghi Hoàng Khoa*

*Ngày báo cáo: 27/03/2024*

**Nhóm: 03 (ghi số thứ tự nhóm)**

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT213.O21.ANTN

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Hà Thị Thu Hiền | 21522056 | 21522056@gm.uit.edu.vn |
| 2 | Phạm Ngọc Thơ | 21522641 | 21522641@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

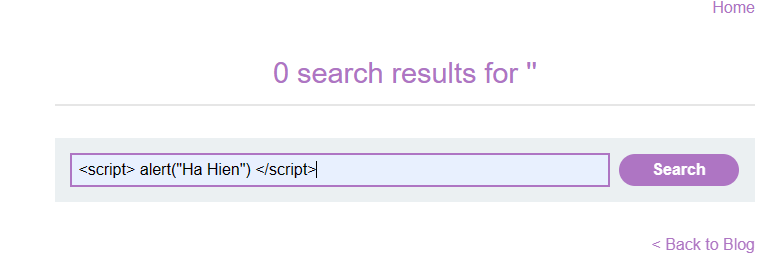
|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** |
| 1 | XSS | 100% |
| 2 | CSRF | 90% |
| 3 | CORS | 50% |

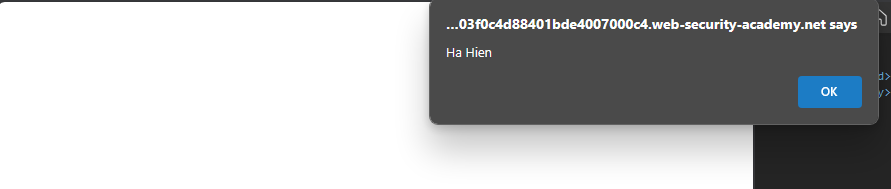
**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

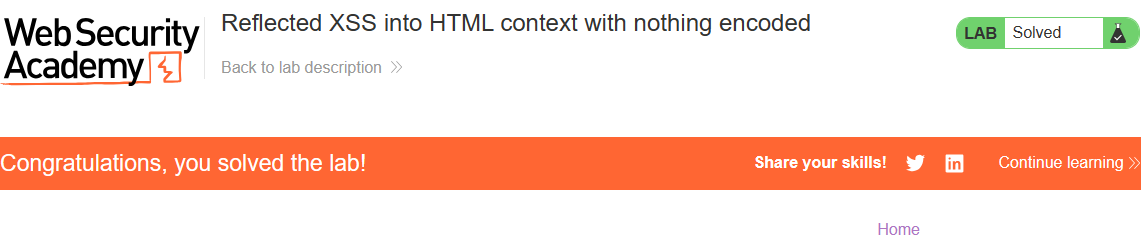
BÁO CÁO CHI TIẾT

1. **Cross-site scripting:**
2. **Reflected XSS into HTML context with nothing encoded**

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

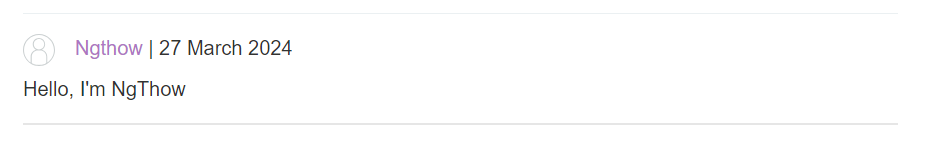




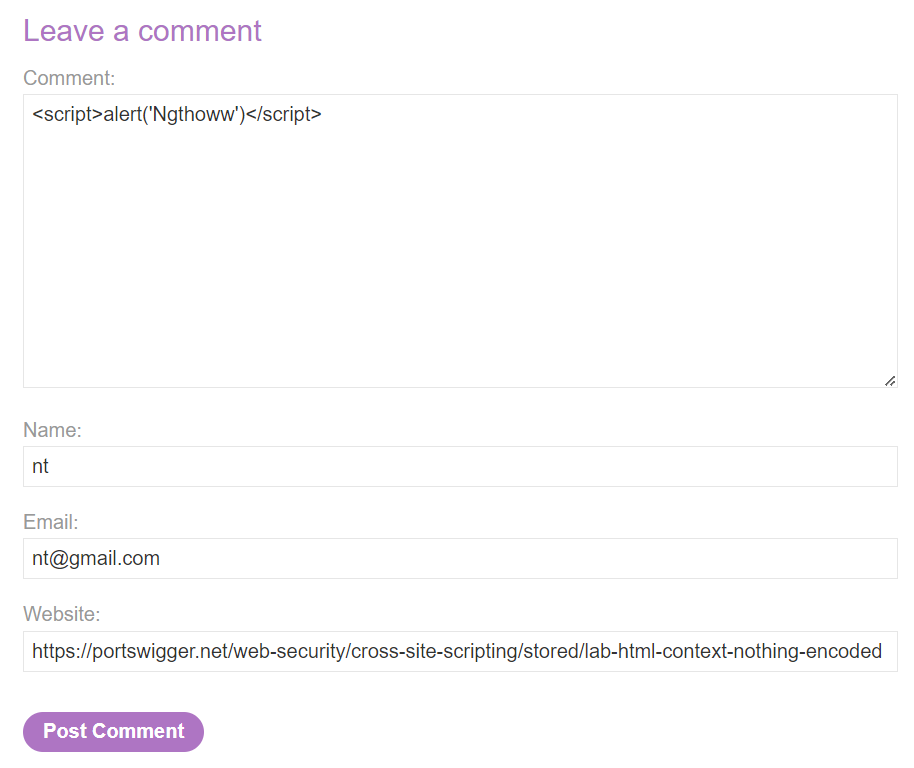


1. **Stored XSS into HTML context with nothing encoded**

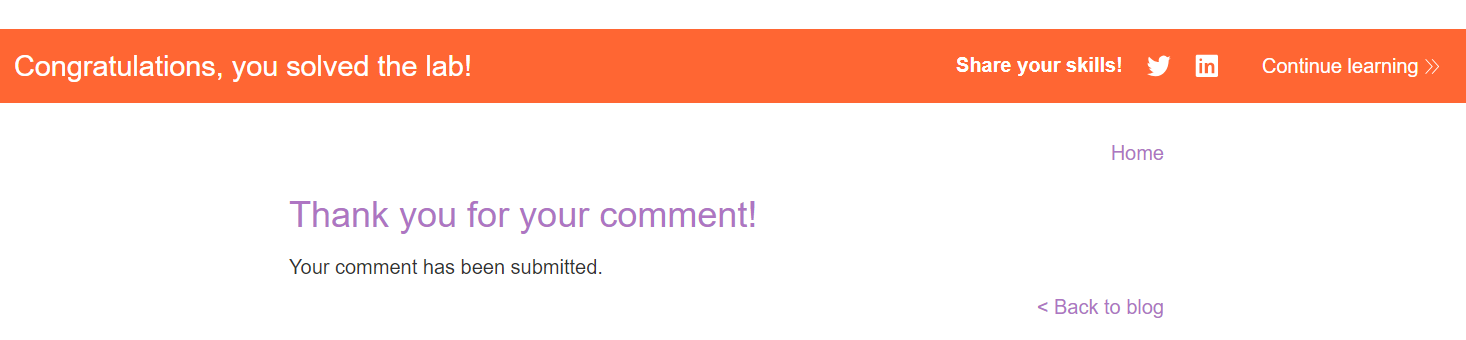
* Đề: Lab này chứa lỗ hổng kịch bản chéo trang được lưu trữ trong chức năng comment. Để giải quyết lab này, hãy gửi comment gọi hàm *alert* khi bài đăng trên blog được xem.
* Do có bug ở phần comment nên em chọn ngẫu nhiên 1 post và viết comment xem thử nó hiển thị như thế nào:



* Đề bài nói rõ là liên quan đến alert, nên em sẽ viết script để chèn vào:

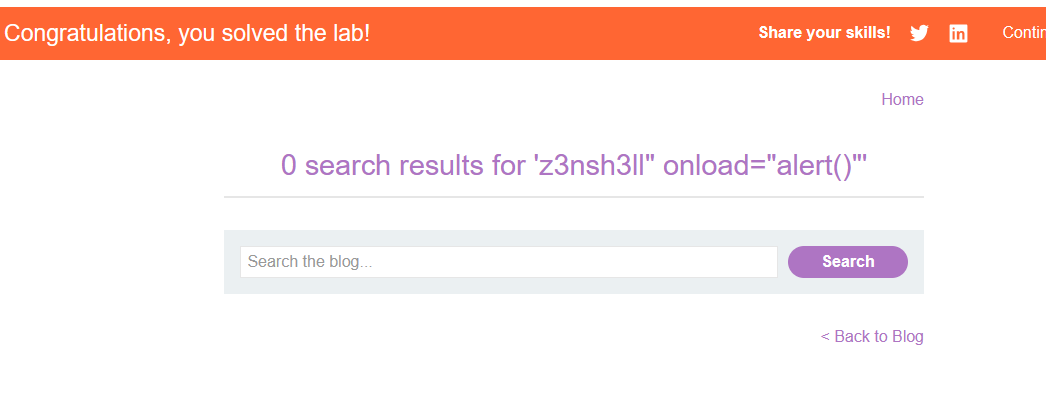


* Kết quả:



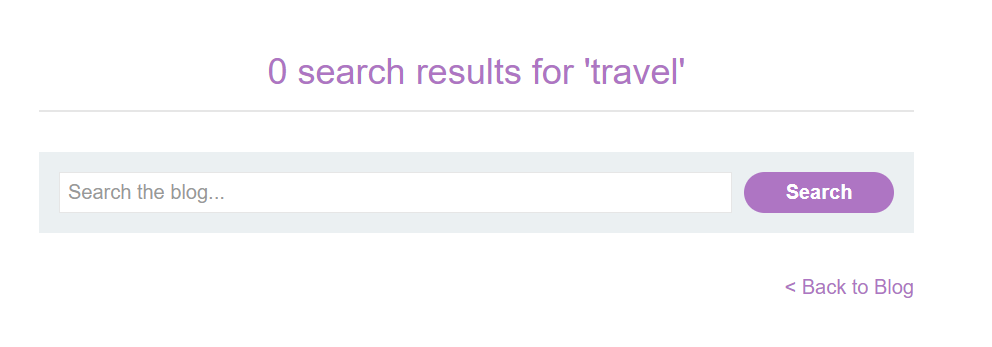
1. **DOM XSS in document.write sink using source location.search**

**Link:** [**https://youtu.be/1f1Jce\_KQUg**](https://youtu.be/1f1Jce_KQUg)

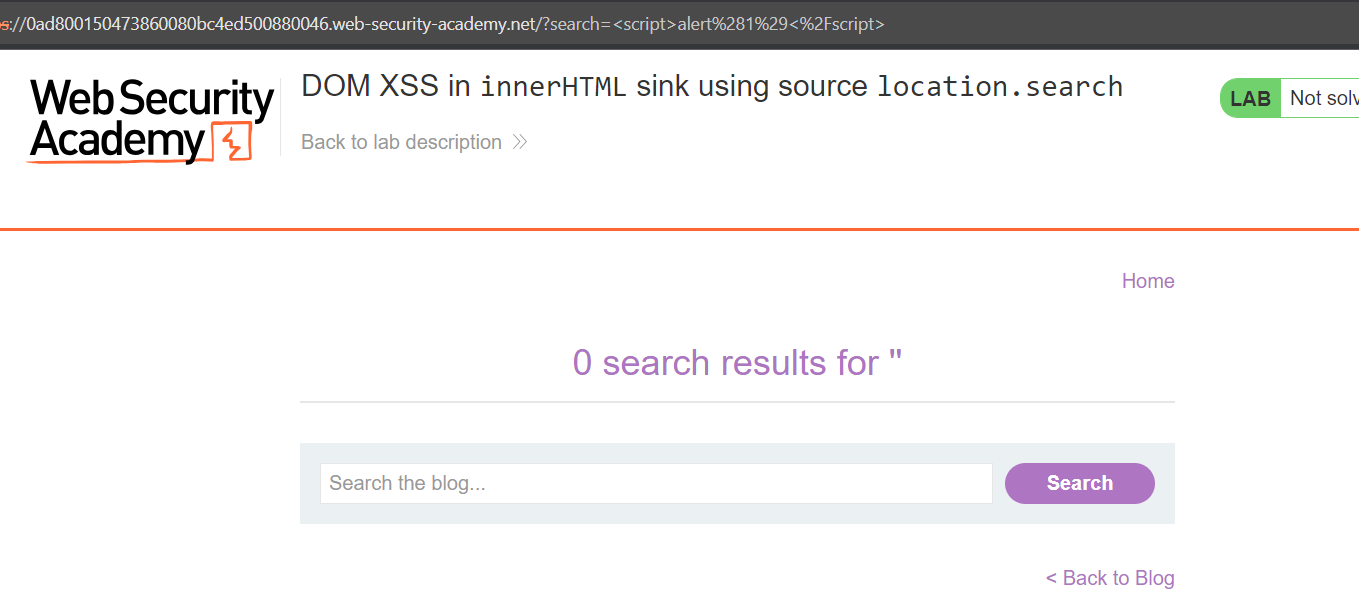


1. **DOM XSS in innerHTML sink using source location.search**

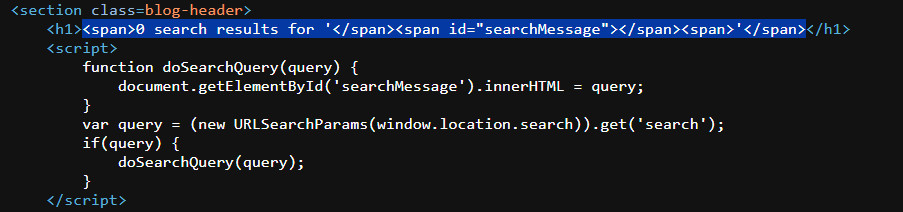
* Đề bài: Lab này chứa lỗ hổng DOM-based cross-site scripting trong chức năng tìm kiếm blog. Nó sử dụng phép gán InnerHTML để thay đổi nội dung HTML của phần tử div, sử dụng dữ liệu từ location.search. Để giải quyết bài thực hành này, hãy thực hiện một cuộc tấn công kịch bản chéo trang gọi hàm cảnh báo.
* Về lỗ hổng DOM-based XSS: thường phát sinh khi JavaScript lấy dữ liệu từ nguồn do kẻ tấn công kiểm soát, chẳng hạn như URL, rồi chuyển dữ liệu đó đến một hệ thống hỗ trợ thực thi mã động, chẳng hạn như eval() hoặc InnerHTML. Điều này cho phép kẻ tấn công thực thi JavaScript độc hại, thường cho phép chúng chiếm đoạt tài khoản của người dùng khác. XSS dựa trên DOM cho phép kẻ tấn công thao túng DOM thông qua việc chèn tập lệnh JavaScript, thường là thông qua dữ liệu do người dùng kiểm soát, thường được tìm thấy trong URL của chính trang web.
* Để giải quyết lab này, đầu tiên em sẽ thực hiện tìm kiếm ngẫu nhiên 1 từ khóa:



* Em đã thử chèn <script>alert(1)</script> nhưng không có hiện được thông báo, và chỉ có URL của trang web thay đổi:



* Em sẽ kiểm tra source code của trang web để xem tại hàm tìm kiếm, chương trình cụ thể đã làm những gì:



Có thể thấy rằng, hàm doSearchQuery() nhận vào tham số query và thay đổi nội dung của searchMessage thành nội dung của query. Code có sử dụng InnerHTML - có thể thêm nội dung bao gồm mã HTML thay vì các chuỗi đơn giản nên điều này có thể gây ra rủi ro bảo mật. Hơn nữa, quan sát giá trị tìm kiếm mà em đã nhập vào khung search, nó sẽ được đặt trong cặp thẻ <span></span>. Khi nãy nhập alert nhưng không có kết quả, lí do có thể là do InnerHTML không thực thi mã JavaScript bên trong thẻ <script>, hoặc nó loại bỏ thẻ <script> hoàn toàn để ngăn chặn bất kỳ hành động độc hại có thể xảy ra. Do đó, em sẽ sử dụng 1 thẻ khác là <img> vốn là thẻ phổ biến trong HTML và kiểu tấn công này.

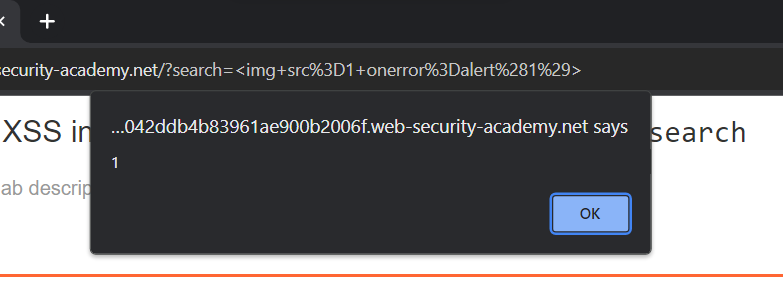
* Thực hiện chèn script như sau:

<img src=1 onerror=alert(1)>

Ở đây, chèn 1 ảnh với đường dẫn source là “1”, tuy nhiên đây là đường dẫn không hợp lệ do đó sẽ kích hoạt một đoạn Javascript độc hại, trong câu lệnh này sẽ kích hoạt trình báo lỗi onerror khi load không được ảnh. Trình báo lỗi gọi đến hàm arlet() để hiển thị hộp thông báo với nội dung là “1”.



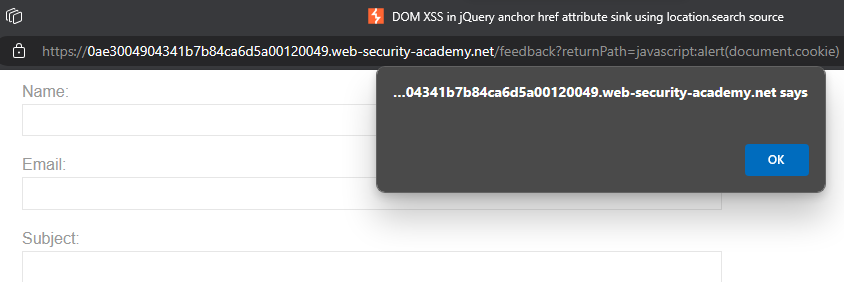
* Kết quả:



1. **DOM XSS in jQuery anchor href attribute sink using location.search source**

**Link:**  [**https://youtu.be/kDdrsZ\_Bj6A**](https://youtu.be/kDdrsZ_Bj6A)



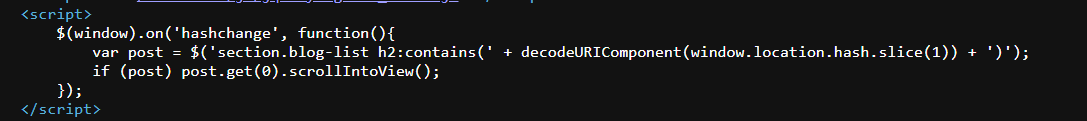


1. **DOM XSS in jQuery selector sink using a hashchange event**

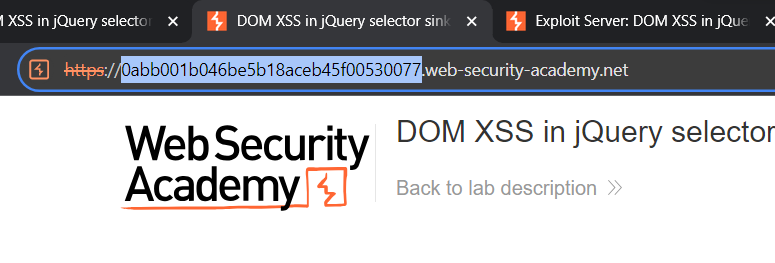
* Đề bài: Lab này chứa lỗ hổng DOM-based cross-site scripting vulnerability trên trang chủ. Nó sử dụng hàm $() selector của jQuery để tự động cuộn đến một bài đăng nhất định, có tiêu đề được chuyển qua thuộc tính location.hash.

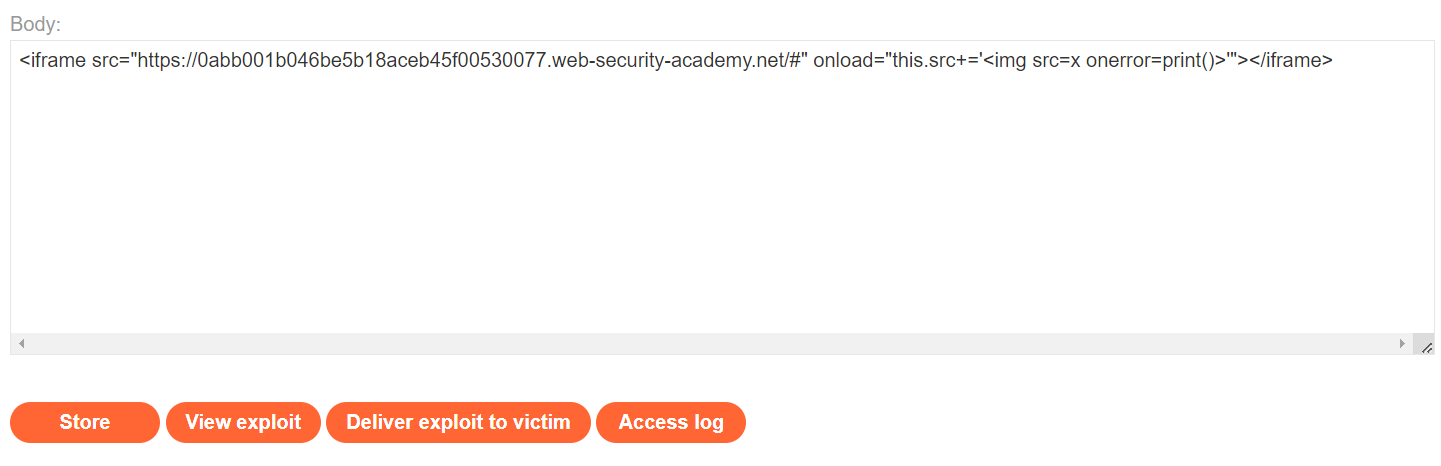
Để giải quyết vấn đề, hãy cung cấp một khai thác cho nạn nhân gọi hàm print() trong trình duyệt của họ.

* Kiểm tra source trang chủ, phát hiện nơi có chứa lỗ hổng liên quan đến thư viện jQuery, biết là jQuery vì nó có ký tự $:



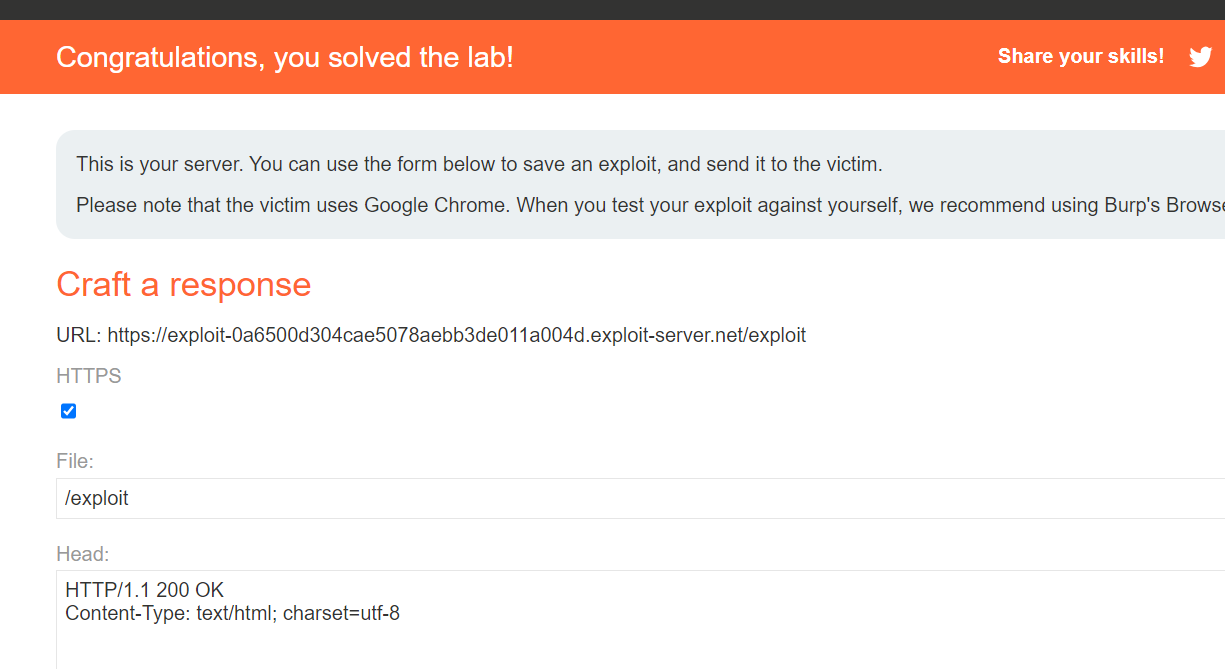
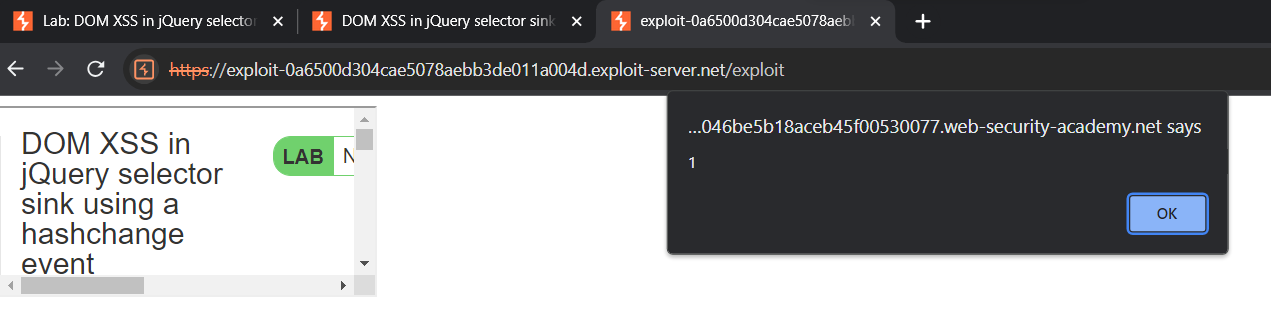
* $(window).on('hashchange', function(){ ... });: Dòng này sử dụng thư viện jQuery để đính kèm một bộ xử lý sự kiện cho sự kiện "hashchange" trên cửa sổ trình duyệt. Sự kiện này xảy ra khi phần hash của URL (phần sau ký tự #) thay đổi. Khi sự kiện này xảy ra, hàm callback được gọi.
* var post = $('section.blog-list h2:contains(' + decodeURIComponent(window.location.hash.slice(1)) + ')');: tìm kiếm một phần tử HTML có cấu trúc như sau: một phần tử <h2> nằm trong một phần tử <section> có lớp CSS là "blog-list" và chứa nội dung văn bản giống với giá trị của phần hash (sau ký tự #) trong URL. Để thực hiện điều này, đoạn mã giải mã phần hash từ URL (sử dụng decodeURIComponent) và sau đó tạo một jQuery selector dựa trên phần hash này.
* if (post) post.get(0).scrollIntoView();: Nếu post tồn tại, thì phần tử đầu tiên trong kết quả (được lưu trong biến jQuery post) được cuộn vào chế độ hiển thị bằng cách sử dụng scrollIntoView(). Điều này sẽ làm cho cửa sổ trình duyệt cuộn đến vị trí của phần tử post.
* Để khai thác lỗ hổng, ta cần đặt <img src=x onerror=alert(document.domain)> vào URL, sau #. Điều này sẽ kích hoạt cảnh báo vì bộ chọn jquery $() trước tiên sẽ cố gắng chọn mục trên trang và khi chuỗi không tồn tại thì nó sẽ thêm element.
* Để giải quyết vấn đề, cần cung cấp một khai thác sao cho nạn nhân gọi hàm print() trong trình duyệt của họ. Tuy nhiên khó để làm được việc này, do đó em sẽ nhúng mã vào trang web hiện tại bằng <iframe> để tự động hiển thị URL mà không cần thao tác từ victim:





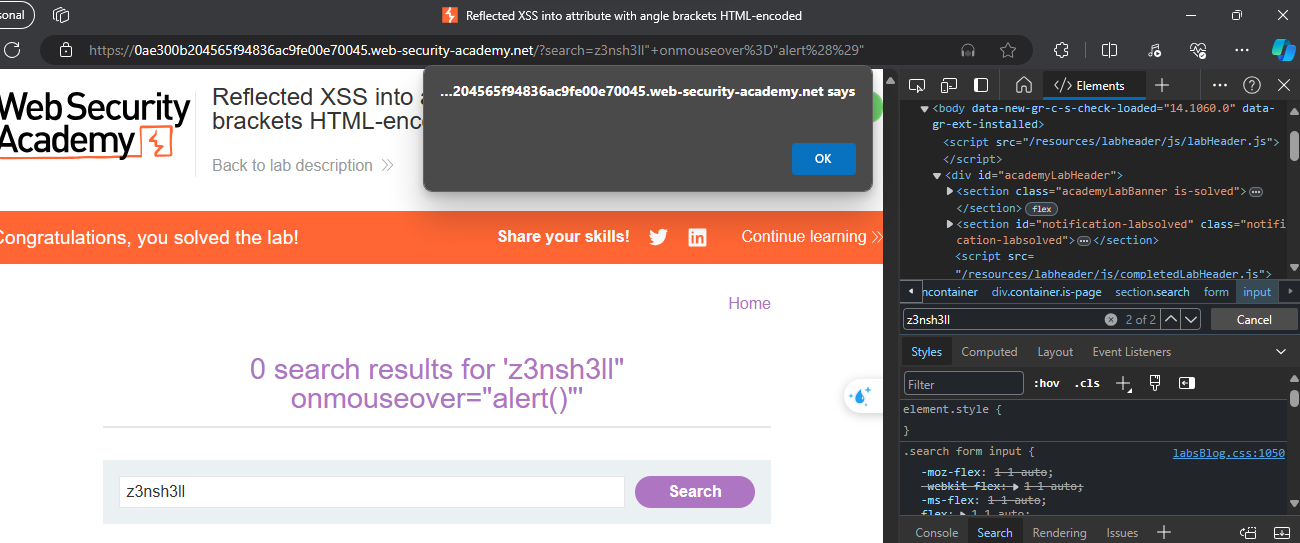
Về payload: src sẽ là URL của trang web, ở đây là trang mà lab trình bày yêu cầu của challenge. Sau khi load xong trang web, sự kiện onload sẽ được kích hoạt, một đoạn mã HTML sẽ được nối vào URL của iframe, cụ thể là chèn một đoạn mã để load hình ảnh có thuộc tính onerror. Đoạn này giống với bài 4 em đã trình bày phía trên ạ.

* Kết quả:



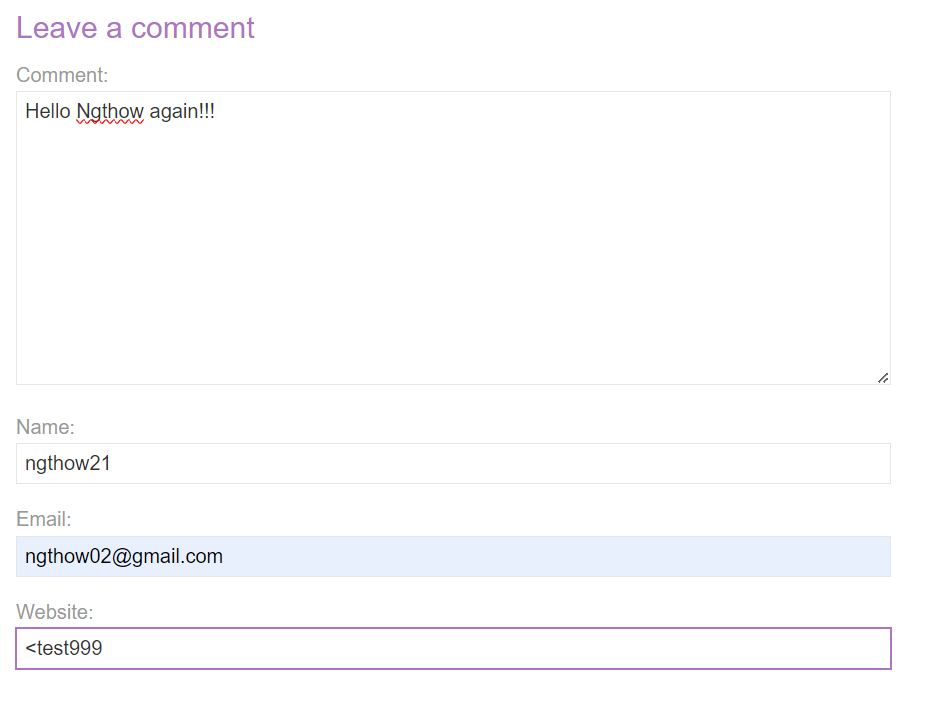
1. **Reflected XSS into attribute with angle brackets HTML-encoded**

**Link:** [**https://youtu.be/48dsZvtzplk**](https://youtu.be/48dsZvtzplk)

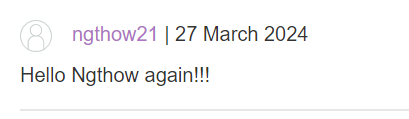


1. **Stored XSS into anchor href attribute with double quotes HTML-encoded**

* Đề bài: Lab này chứa lỗ hổng kịch bản chéo trang được lưu trữ trong chức năng nhận xét. Để giải bài thực hành này, hãy gửi nhận xét gọi hàm cảnh báo khi nhấp vào tên tác giả nhận xét.
* Đầu tiên, em sẽ comment:



* Hiển thị trên phần bình luận như sau:

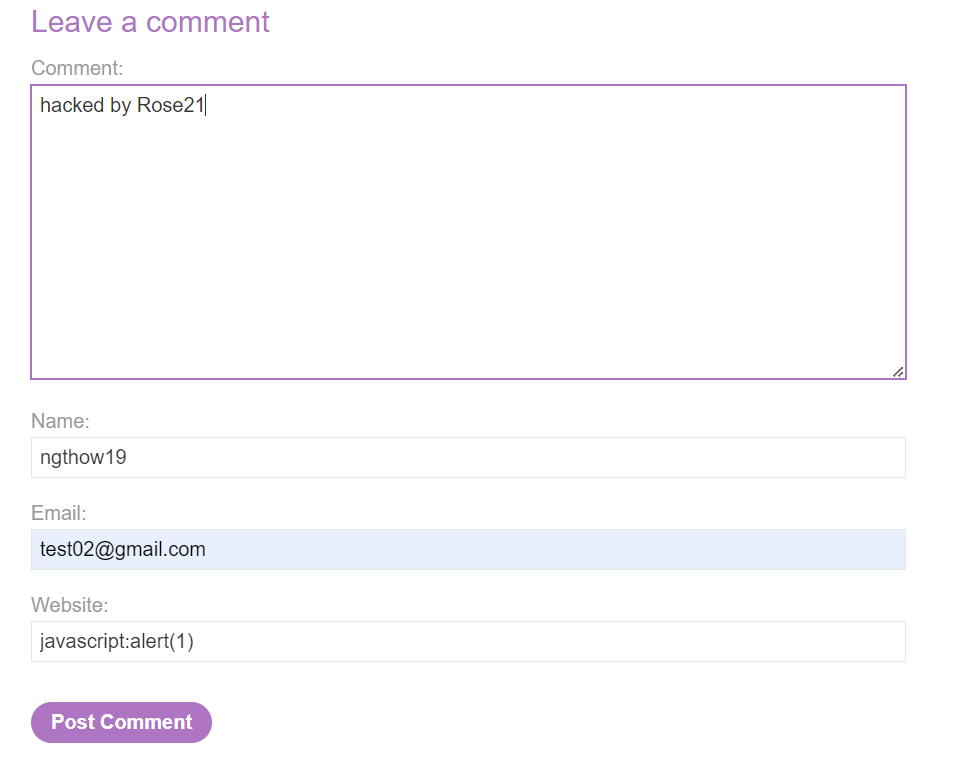


* Kiểm tra source web xem nó xử lý dữ liệu em nhập vào như thế nào, lưu ở đâu:

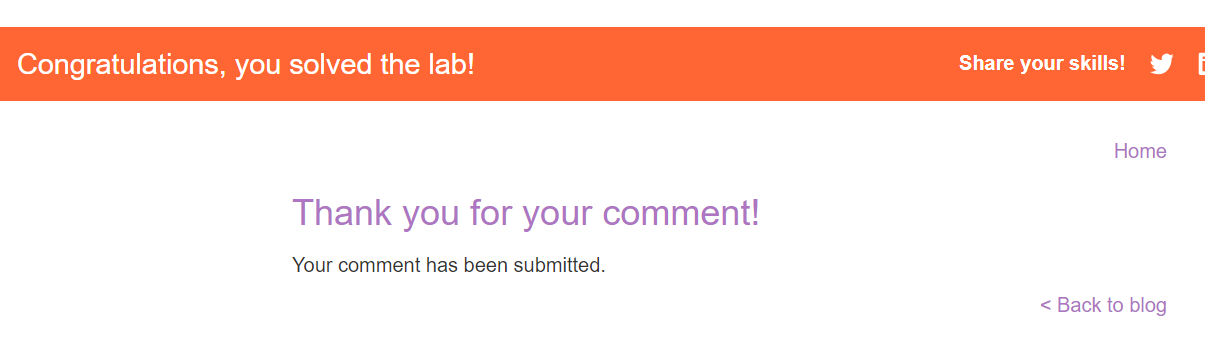


* Chú ý mục author, điều đáng nói là khi điền trang web, em không điền đầy đủ với cú pháp http://... mà chỉ đơn giản là <test999, nhưng chương trình vẫn nhận và lưu vào href, dường như nó chỉ cần chứa chữ và ký tự đặc biệt là đủ, đây sẽ là điểm chúng ta lợi dụng để tiêm đoạn mã mong muốn. Do đó em sẽ điền vào mục Website nội dung sau - là một lệnh phổ biến để khai thác XSS:

javascript:alert(1)



* Kết quả:

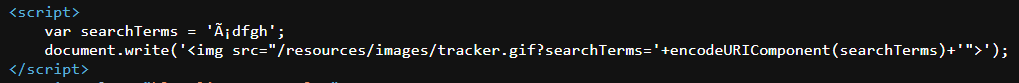


1. **Reflected XSS into a JavaScript string with angle brackets HTML encoded**

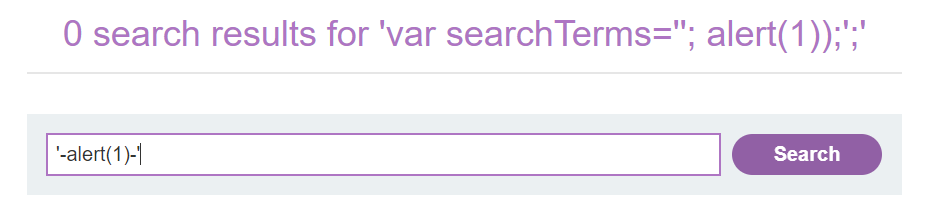
* Đề bài: This lab contains a reflected cross-site scripting vulnerability in the search query tracking functionality where angle brackets are encoded. The reflection occurs inside a JavaScript string. To solve this lab, perform a cross-site scripting attack that breaks out of the JavaScript string and calls the alert function.
* Tìm kiếm tại trang chủ:

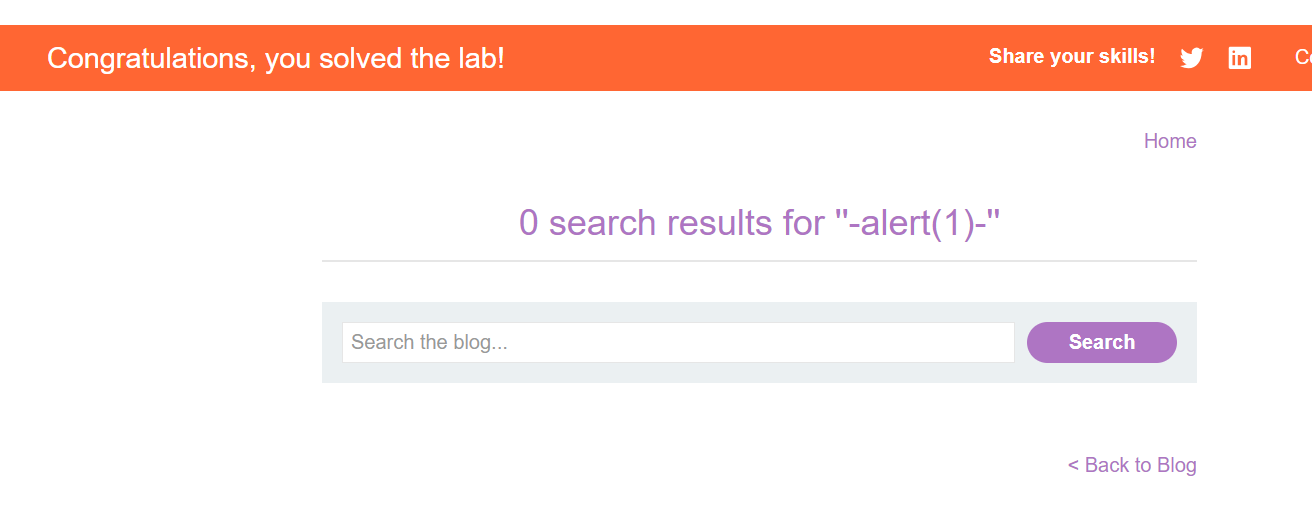


* Kiểm tra source code:



* Quan sát document.write, tất cả các ký tự đều được mã hóa bằng EncodeURIComponent, vì vậy XSS sẽ không hoạt động được.
* Chúng ta cần tìm hiểu xem kiểu dữ liệu nào được mã hóa:
* Thử nhập lần lượt các payload đặc biệt, và đến ‘alert(1)-’ thì thành công:



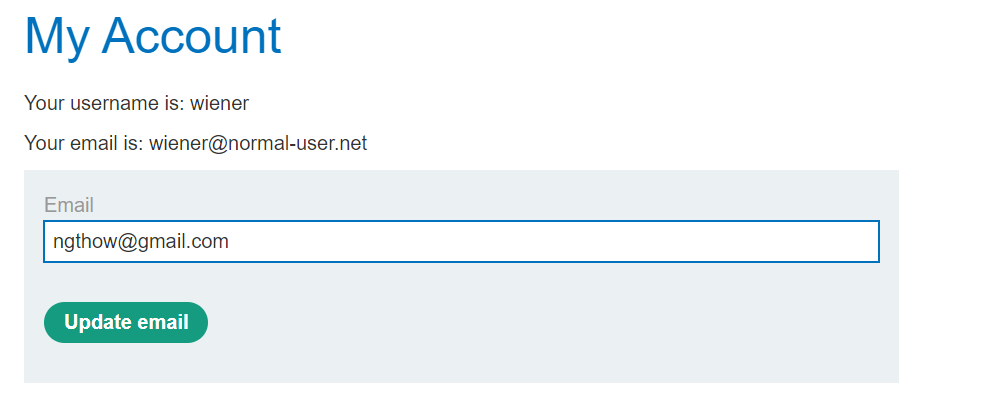


1. **CSRF:**
2. **CSRF vulnerability with no defenses**

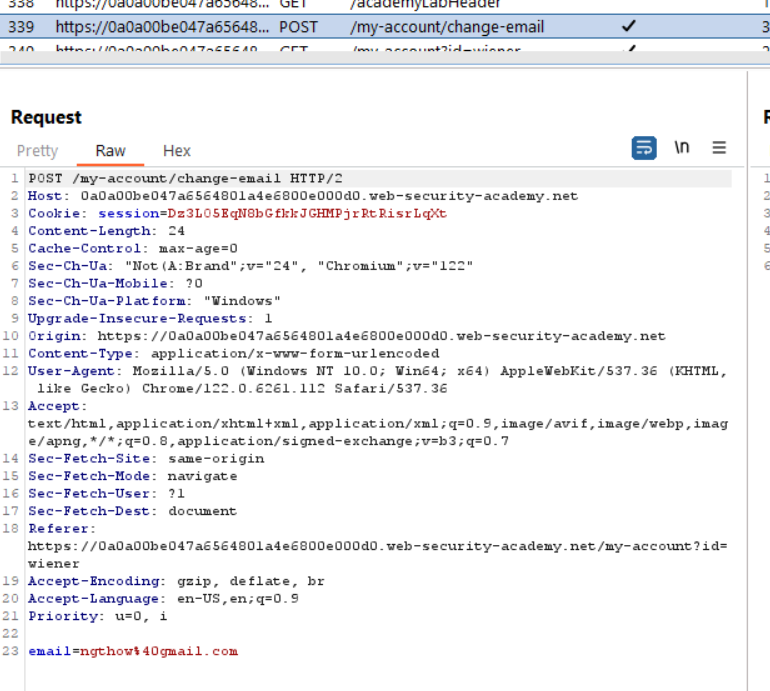
* Đề bài: Chức năng thay đổi email của lab này dễ bị tấn công bởi CSRF.

Để giải quyết bài lab, hãy tạo một số HTML sử dụng tấn công CSRF để thay đổi địa chỉ email của người xem và tải nó lên máy chủ khai thác của bạn.

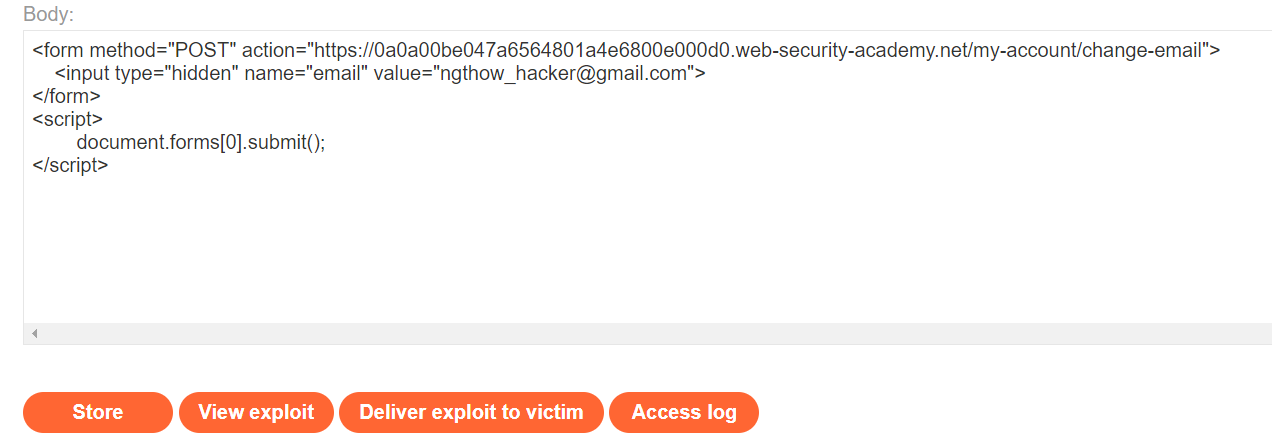
* Đầu tiên em sẽ sửa lại email thành ngthow@gmail.com:



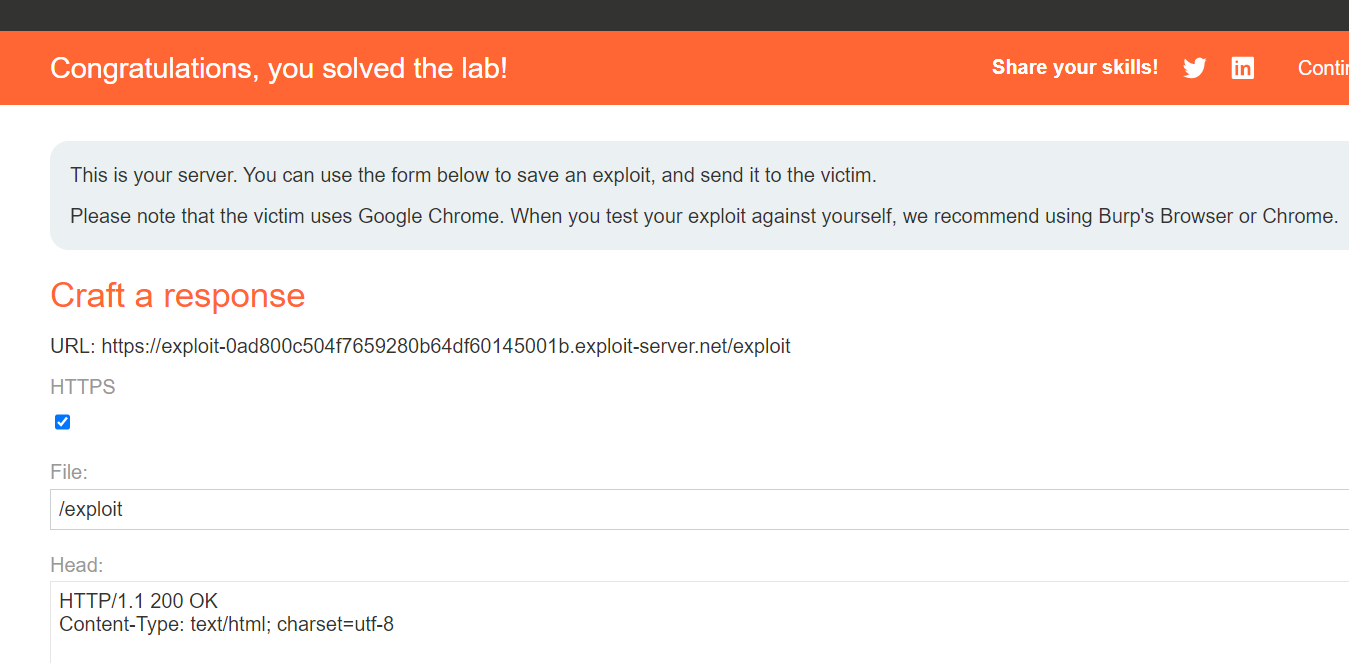
* Kiểm tra bên HTTP history để xem chi tiết gói tin request đổi email, có thể thấy rằng trang web nhận địa chỉ email mới thông qua tham số email:



* Copy URL để chèn vào template HTLM để khai thác lỗ hổng, ở đây sửa email thành email khác cái đã điền ban đầu “ngthow\_hacker@gmail.com”:



* “Deliver exploit to victim” để chuyển đến máy nạn nhân, kết quả thành công:



1. **CORS:**
2. **CORS vulnerability with basic origin reflection**
3. **CORS vulnerability with trusted null origin**

* Đề bài: Trang web này có cấu hình CORS không an toàn ở chỗ nó tin tưởng vào nguồn gốc "null". Để giải quyết bài thí nghiệm, hãy tạo một số JavaScript sử dụng CORS để truy xuất khóa API của quản trị viên và tải mã lên máy chủ khai thác của bạn. Lab sẽ được giải quyết khi bạn gửi thành công khóa API của quản trị viên. Bạn có thể đăng nhập vào tài khoản của mình bằng thông tin đăng nhập sau: wiener:peter
* Sơ lược về CORS: đây là thông tin em đọc được trên Viblo ạ!

Trong trường hợp đơn giản nhất, phía client sẽ tạo request GET, POST, PUT, HEAD,... để yêu cầu server làm một việc gì đó. Những request này sẽ được đính kèm một header tên là Origin để chỉ định origin của client code (giá trị của header này chính là domain của trang web).

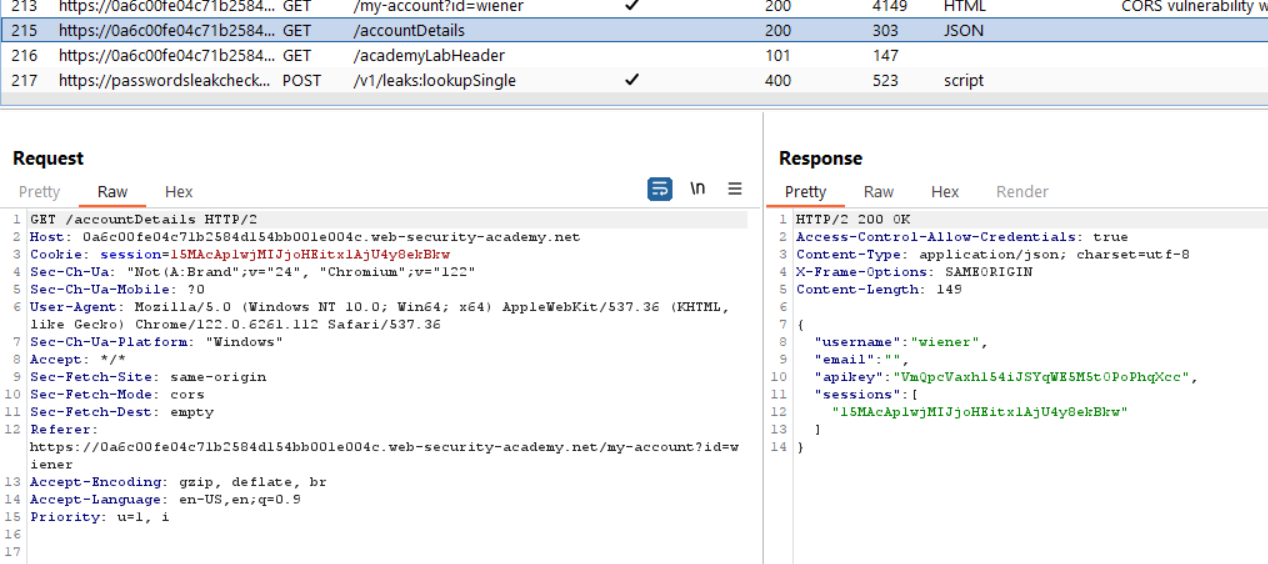
Server sẽ xem xét Origin để biết được nguồn này có phải là nguồn hợp lệ hay không. Nếu hợp lệ, server sẽ trả về response kèm với header Access-Control-Allow-Origin. Header này sẽ cho biết xem client có phải là nguồn hợp lệ để browser tiếp tục thực hiện quá trình request.

Trong trường hợp thông thường, Access-Control-Allow-Origin sẽ có giá trị giống như Origin, một số trường hợp giá trị của Access-Control-Allow-Origin sẽ nhìn giống giống như Regex hay chỉ đơn giản là \*, tuy nhiên thì cách dùng \* thường được coi là không an toàn, ngoại trừ trường hợp API của bạn được public hoàn toàn và ai cũng có thể truy cập được.

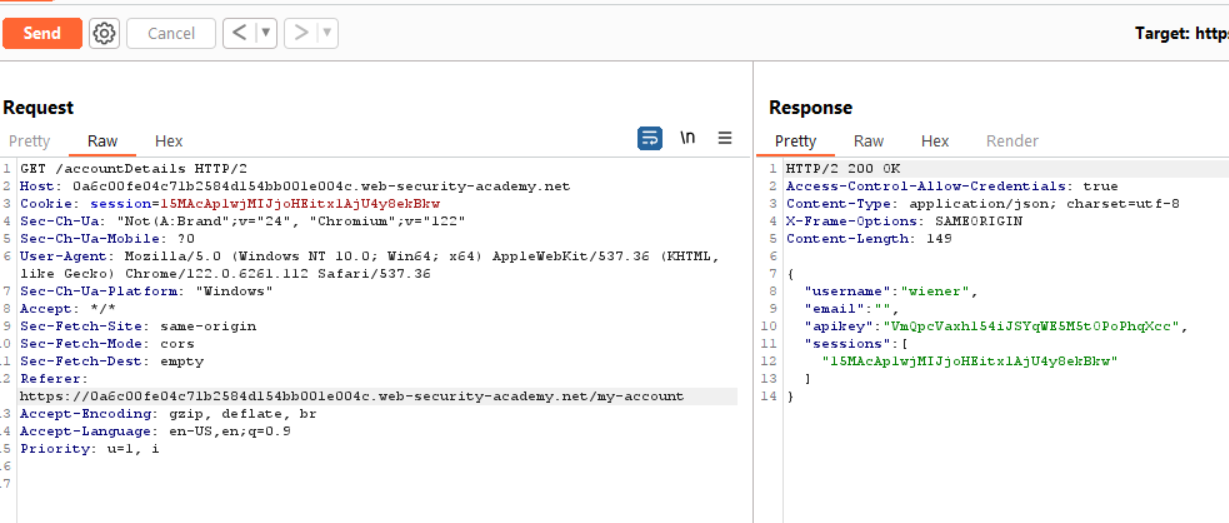
* Sau đây, em sẽ tiến hành thực hiện challenge. Đầu tiên, đăng nhập để lấy API key:



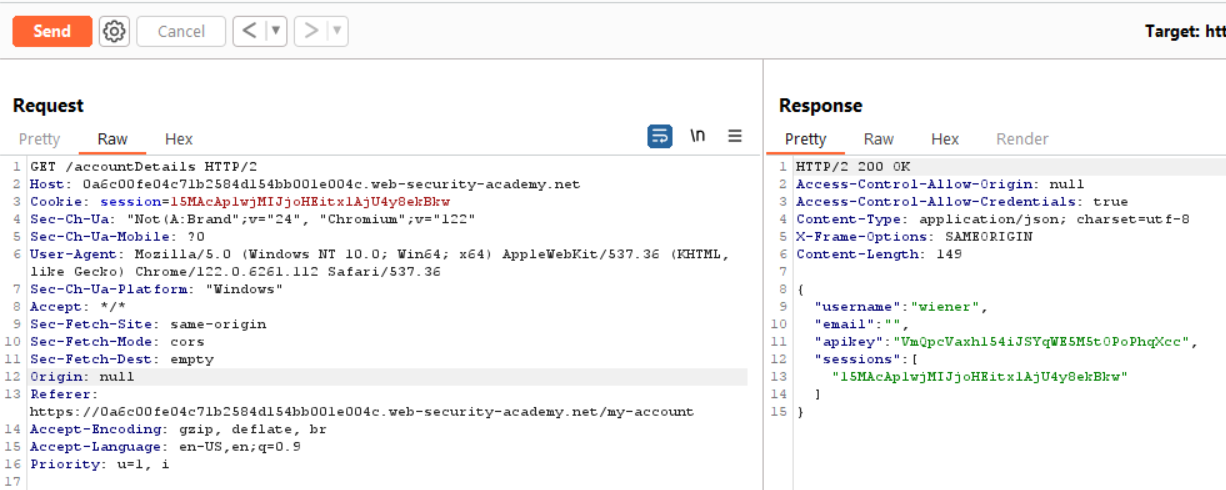
* Sử dụng HTTP history để xem request:



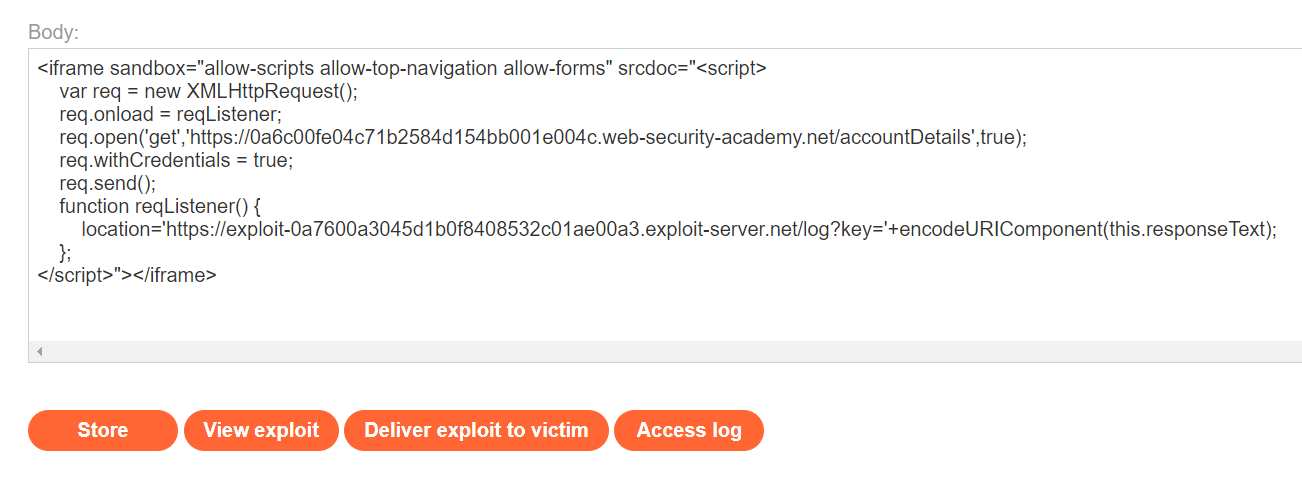
* Chỉnh sửa request, xóa ở trường Referer ?id=wiener, vẫn nhận response bình thường:



* Thêm trường origin: null, respones cũng đã cập nhật Access-Control-Allow-Origin: null như hình:



* Em cũng đã làm theo hướng dẫn solution của lab, tuy nhiên kết quả không khả quan ạ!



---

***Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này***

# **YÊU CẦU CHUNG**

* Sinh viên tìm hiểu và thực hành theo hướng dẫn.
* Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
* Sinh viên báo cáo kết quả thực hiện và nộp bài.

**Báo cáo:**

* File .DOCX và .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
* Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
* Đặt tên theo định dạng: [Mã lớp]-SessionX\_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành đã đăng ký với GVHD-TH).

*Ví dụ: [*NT101.K11.ATCL*]-Session1\_Group3.*

* Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
* Không đặt tên đúng định dạng – yêu cầu, sẽ **KHÔNG** chấm điểm bài Lab.
* Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá**: Sinh viên hiểu và tự thực hiện được bài thực hành. Khuyến khích:

* Chuẩn bị tốt.
* Có nội dung mở rộng, ứng dụng trong kịch bản phức tạp hơn, có đóng góp xây dựng bài thực hành.

*Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)