

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO CUỐI KÌ MÔN NHẬP MÔN BẢO MẬT THÔNG TIN

Người hướng dẫn: **GV. HUỲNH NGỌC TÚ**

Người thực hiện: **TRẦN THỊ VỆ – 53200674**

NGUYỄN ĐÌNH DANH - 52100878

Lớp : 21050301

Khoá : 25

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO CUỐI KÌ MÔN NHẬP MÔN BẢO MẬT THÔNG TIN

Người hướng dẫn: **GV. HUỲNH NGỌC TÚ**

Người thực hiện: **TRẦN THỊ VỆ – 53200674**

NGUYỄN ĐÌNH DANH - 52100878

Lớp : 21050301

Khoá : 25

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin chân thành gửi lời cảm ơn và sự tri ân sâu sắc đến với các thầy cô, giảng viên của khoa Công nghệ thông tin nói chung và cô Huỳnh Ngọc Tú nói riêng. Trong suốt quá trình học tập và rèn luyện, chúng em đã nhận được rất nhiều sự giúp đỡ tận tình, sự quan tâm, chăm sóc của các thầy cô.

Ngoài ra, chúng em còn được cô Huỳnh Ngọc Tú, cô đã dạy bảo những kiến thức, phương pháp mới về bảo mật và mã hóa hay ho và thú vị... Chúng em xin cảm ơn các cô rất nhiều trong suốt quá trình học tập này ạ!!!

Bởi lượng kiến thức của chúng em còn hạn hẹp và gặp nhiều vấn đề trong quá trình học nên báo cáo này sẽ còn nhiều thiếu sót và cần được học hỏi thêm. Em rất mong em sẽ nhận được sự góp ý của quý thầy cô về bài báo cáo này của em để em rút kinh nghiệm trong những môn học sắp tới. Cuối cùng, em xin chân thành cảm ơn quý thầy cô.

TP Hồ Chí Minh, ngày 01 tháng 11 năm 2022

Trần Thị Vẹn

Nguyễn Đình Danh

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của GV. Huỳnh Ngọc Tú. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 01 tháng 12 năm 2022

Tác giả

(ký tên và ghi rõ họ tên)

Trần Thị Vẹn

Nguyễn Đình Danh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Trong bài báo cáo này chúng em sẽ áp dụng các kiến thức mà giảng viên đã cung cấp trong quá trình giảng dạy để giải quyết các câu hỏi cũng như vấn đề đã được đặt ra trong đề báo cáo . Thêm vào đó, sử dụng các phần mềm để hỗ trợ bài báo cáo này hoàn thiện và chính xác hơn, ngoài ra còn có kiến thức của phần báo có giữa kì.

Bài báo cáo cả sự quan sát và thực hành nên đòi hỏi sinh viên cần phải vững nắm vững kiến thức về phần mềm để có thể suy luận và kết hợp lại với nhau nên đôi khi sẽ khó tránh khỏi sai sót nhưng chúng em sẽ cố gắng hoàn thành thật tốt đề tài này thật tốt.

MỤC LỤC

LỜI CẢM ƠN	i
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN	iii
TÓM TẮT	iv
MỤC LỤC	1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ	3
I. SQL INJECTION (TẤN CÔNG SQL).....	6
1. Lý thuyết về tấn công SQL	6
1.1. Khái niệm tấn công SQL:	6
1.2. Mục đích tấn công.....	6
1.3. Cách thức hoạt động	7
1.4. Các kiểu tấn công.....	7
1.4.1. Inband SQLI	7
1.4.2. Inferential (Blind) SQLI	9
1.4.3. Out of band SQLi	11
1.5. Cách phòng chống SQL injection.....	11
2. Demo về SQL injection	13
2.1. Giải thích lý thuyết về code demo:	13
2.2. Demo và chạy thử SQL injection:	19
II. XSS INJECTION (TẤN CÔNG XSS)	23
1. Lý thuyết về tấn công XSS	23
1.1. Cross site scripting – XSS là gì?.....	23
1.2. Mục đích thực hiện cuộc tấn công XSS.....	24
1.3. Các bước thực hiện tấn công XSS	24
1.4. Các loại tấn công XSS	24
1.4.1. Reflected XSS (non-persistent XSS).....	24
1.4.2. Stored XSS (persistent XSS).....	25

1.4.3.	DOM-based vulnerabilities	26
1.5.	Các biện pháp phòng chống tấn công XSS.....	28
1.5.1.	Lọc (XSS Filter)	28
1.5.2.	Thoát khỏi XSS (XSS Escape).....	29
1.5.3.	Đặt cờ HttpOnly	29
1.5.4.	Sử dụng CSP.....	30
2.	Phần mềm cũng như cài đặt hệ thống demo XSS.....	31
3.	Demo các loại tấn công XS.....	37
3.1.	Demo Reflected XSS	38
3.1.1.	Mức độ an toàn thấp – Security Low.....	39
3.1.2.	Mức độ an toàn trung bình – Security Medium.....	40
3.1.3.	Mức độ an toàn cao – Security High	42
3.2.	Demo Stored XSS	43
3.2.1.	Mức độ an toàn thấp – Security Low.....	43
3.2.2.	Mức độ an toàn trung bình và cao – Security Medium and High.....	45
3.3.	Demo Dom - based	46
3.3.1.	Mức độ an toàn thấp – Security Low.....	48
3.3.2.	Mức độ an toàn trung bình và cao– Security Medium and High.....	51

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 1: Mô tả SQL Injection	6
Hình 2: Mô tả cách thức hoạt động của SQL Injection.	7
Hình 3: Input lỗi	8
Hình 4: Hệ thống phát sinh lỗi do input	8
Hình 5: URL mẫu	9
Hình 6: Đoạn mã nhập vào username	10
Hình 7: Đoạn mã nhập vào username	10
Hình 8: Đoạn mã nhập vào username	10
Hình 9: Đoạn mã nhập vào username	10
Hình 10: Ví dụ về regex	12
Hình 11: Mô tả đặc quyền cho người dùng bình thường	13
Hình 12: Bảng “users” để chứa dữ liệu username và password	14
Hình 13: File config	14
Hình 14: File login	15
Hình 15: Xử lý input	15
Hình 16: Chuyển hướng đăng nhập	16
Hình 17: Không tìm thấy user và pass thì kết nối tới database	16
Hình 18: CÁC BƯỚC THỰC HIỆN TẤN CÔNG REFLECTED XSS	25
Hình 19: CÁC BƯỚC THỰC HIỆN TẤN CÔNG STORED XSS	26
Hình 20: Các loại source và các kiểu thể hiện	27
Hình 21: Các loại sink và các kiểu thể hiện	27
Hình 22: CÁC BƯỚC THỰC HIỆN TẤN CÔNG DOM-BASED XSS	28
Hình 23: Ví dụ về 5 ký tự được chuyển đổi của HTML	29
Hình 24: Mở Xampp và bấm Start với 2 mục đầu	32
Hình 25: Giải nén tệp htdocs từ file DVWA đã tải	32

Hình 26: Đổi tên file và thể loại đuôi file config	33
Hình 27: Nhấn Create/Reset Database	34
Hình 28: Lỗi xuất hiện	34
Hình 29: Chỉnh sửa lại 2 dòng trong file Notepad	35
Hình 30: Nhấn lại Create/Reset Database	35
Hình 31: Nút login ở cuối	36
Hình 32: Hình ảnh form login hiện lên	36
Hình 33: Giao diện bạn đã tải xong phần mềm	37
Hình 34: Thiết lập DVWA Security theo các mức độ	37
Hình 35 Demo nhập môn bảo mật thông tin	38
Hình 36: Hiển thị kết quả của demo	39
Hình 37: Source code ở mức độ thấp	39
Hình 38: Chèn script vào ô input	40
Hình 39: Kết quả sau khi chèn	40
Hình 40: Source code ở mức độ trung bình	41
Hình 41: Khi dùng script mã hóa không được	41
Hình 42: Insert đoạn script ở dạng khác	41
Hình 43: Kết quả vẫn bị ăn cắp cookie	42
Hình 44: Source code ở mức độ an toàn cao	42
Hình 45: Chèn thẻ img ở high level	42
Hình 46: Thành công lấy được cookie	43
Hình 47: Nhập demo đúng	43
Hình 48: Demo đúng và xuất hiện thông điệp	43
Hình 49: Source code của mức độ an toàn thấp	44
Hình 50: Chèn thẻ script vào để demo stored xss	44
Hình 51: Thẻ <h3> đã làm thay đổi cỡ chữ của localhost này	45
Hình 52: Source code mức độ cao	45

Hình 53: Chèn thẻ img vào trong ô name	46
Hình 54: Lấy được cookie thông qua thẻ img	46
Hình 55: Trước khi click chuột chọn select	47
Hình 56: Sau khi chọn select thì address bar cũng thay đổi set chế độ default	48
Hình 57: Source code của mức độ an toàn thấp	48
Hình 58: Chèn script vào address bar	49
Hình 59: Sau khi đã nhấn enter và có được 1 ngôn ngữ mới xuất hiện	49
Hình 60: Source code ở mức độ thấp	50
Hình 61: Lấy được cookie	51
Hình 62: Source code mức độ trung bình	51
Hình 63: Source code mức độ cao	51
Hình 64: Chèn dòng script lên trên thanh địa chỉ	52
Hình 65: Lấy được cookie ở mức độ cao	52

I. SQL INJECTION (TẤN CÔNG SQL)

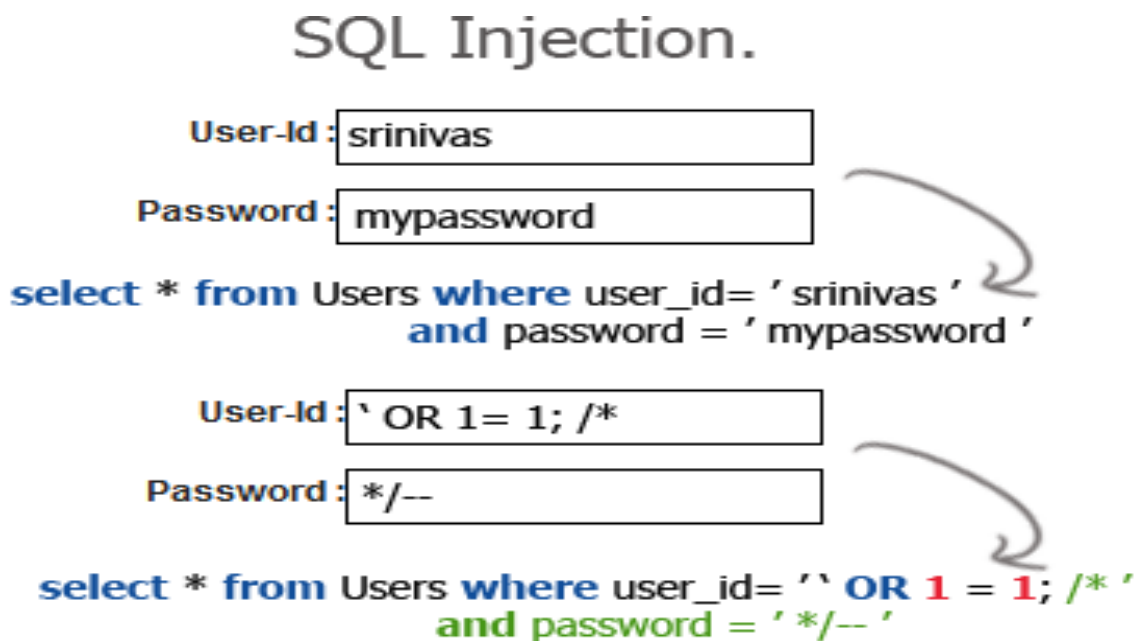
1. Lý thuyết về tấn công SQL

1.1. *Khái niệm tấn công SQL:*

SQL Injection là một hình thức tấn công bằng cách chèn thêm các đoạn mã truy vấn “độc hại” trong chuỗi dữ liệu được nhập vào từ người dùng, sau đó nó được gửi tới SQL Server để phân tích và thực thi.

Điều này gây ra tác hại rất lớn, do tin tặc có thể có toàn quyền sử dụng và thay đổi toàn bộ hệ cơ sở dữ liệu.

Do đó, đây là một trong những cách tấn công vào các kho lưu trữ dữ liệu đơn giản và hiệu quả nhất.



Hình 1: Mô tả SQL Injection

1.2. *Mục đích tấn công*

Tin tặc có thể sử dụng SQL Injection với rất nhiều mục đích khác nhau, ví dụ như:

- Xác định lược đồ cơ sở dữ liệu.

- Trích xuất dữ liệu.
- Thêm, sửa, xoá các bảng hoặc dữ liệu.
- Bỏ qua khâu xác thực.
- Chiếm quyền điều khiển hệ thống.
- Thực thi các lệnh từ xa.

1.3. Cách thức hoạt động

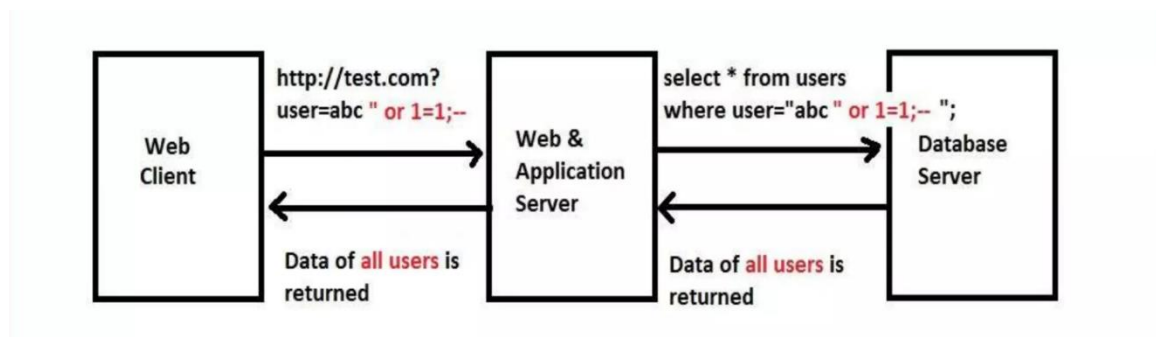
Bước 1: App gửi form điền thông tin cho người dùng.

Bước 2: Tin tặc điền vào form các đoạn mã SQL có thể khai thác thông tin.

Bước 3: App thu nhận thông tin từ form và tổng hợp nó thành dạng string, sau đó gửi các lệnh truy vấn thông tin cần thiết đến database.

Bước 4: Database thực hiện truy vấn, bao gồm cả các lệnh bên trên, sau đó gửi về app.

Bước 5: App trả về thông tin cho người dùng.



Hình 2: Mô tả cách thức hoạt động của SQL Injection.

1.4. Các kiểu tấn công

1.4.1. Inband SQLI

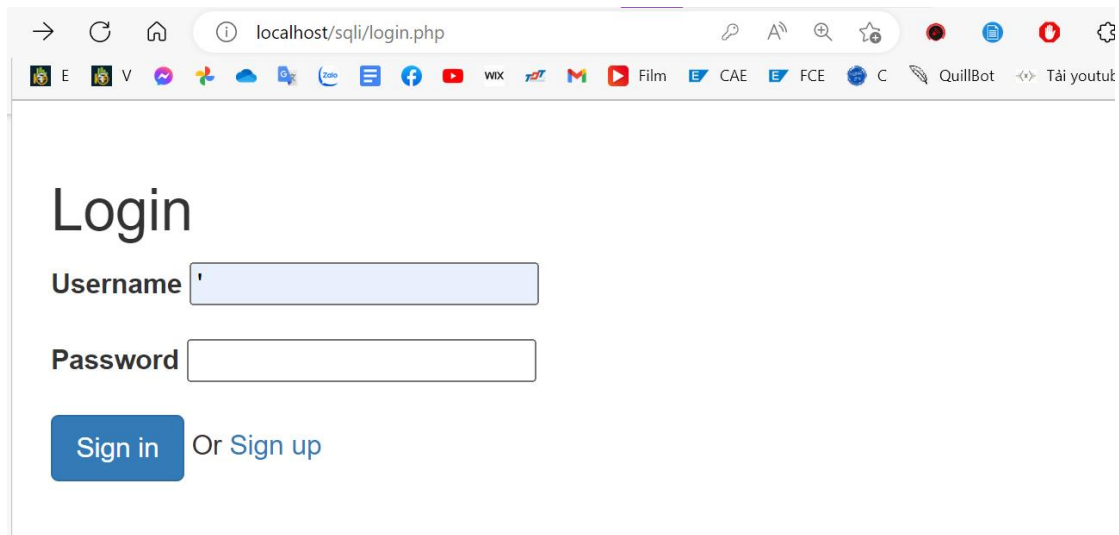
Đây là kiểu tấn công SQL thường thấy nhất và đơn giản nhất để khai thác thông tin. “In-band SQLI” nghĩa là khi kẻ tấn công thực hiện việc chèn mã và trực tiếp nhận về kết quả trên cùng một kênh (ví dụ như tại trang web mà kẻ tấn công nhập mã độc).

2 dạng thường thấy của In-band SQLI là Error-base SQLI và UNION-base SQLI. In-Band SQLi gồm 2 loại chính:

1.4.1.1. Error-based SQLi

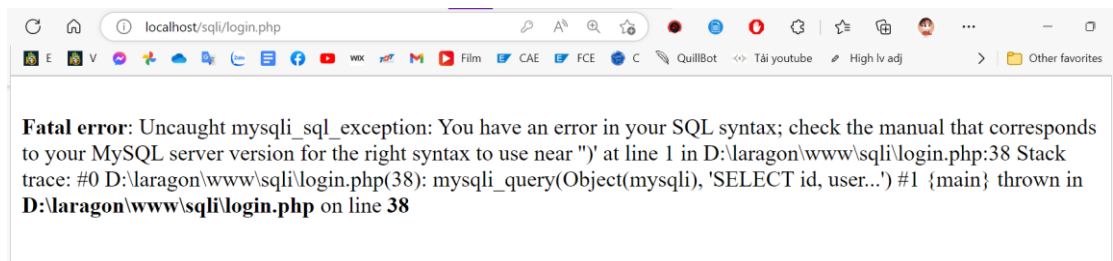
Trong Error-based SQLi, kẻ tấn công sẽ cố gắng nhập mã độc “lỗi”, đoạn mã này khi kết hợp với câu truy vấn sẽ làm hệ thống sinh ra lỗi, lỗi này sẽ cung cấp thông tin về cấu trúc của database, đó sẽ là cơ sở cho các cuộc tấn công sau này.

Ví dụ với form đăng nhập sau, người dùng nhập vào dấu nhảy đơn:



Hình 3: Input lỗi

Thì khi tổng hợp thành câu truy vấn, dấu nhảy đơn bị dư ra sẽ làm hệ thống phát sinh lỗi.



Hình 4: Hệ thống phát sinh lỗi do input

Từ lỗi như trên hình, chúng ta có thể khai thác được những thông tin cơ bản bên trong như:

- Database của web sử dụng là Mysql.
- Lỗi này xảy ra do dấu nhảy đơn bị dư đã làm phát sinh lỗi trong khi tổng hợp câu lệnh truy vấn.

- Lỗi nằm ở dòng số 38, file login.php ...v.v...

1.4.1.2. Union-based SQLi

Cách tấn công này sử dụng toán tử UNION trong ngôn ngữ truy vấn để tổng hợp kết quả của 2 hoặc nhiều câu truy vấn SELECT từ nhiều bảng khác nhau trong database.

1.4.2. Inferential (Blind) SQLi

Đây cũng là một kiểu tấn công tương đối phổ biến. Nó được gọi là tấn công “suy diễn” vì tin tặc sẽ không trực tiếp nhận được kết quả ngay trên cùng một kênh như kiểu Inband, mà phải phân tích dần dần để thu được kết quả cuối cùng.

Inferential SQLi diễn ra chậm hơn nhiều do còn phải quan sát và phân tích database dựa vào cách phản hồi, kết quả trả về từ máy chủ, nhưng nhìn chung độ thiệt hại thì không khác gì Inband.

- Inferential SQLi chia làm hai kiểu chính:
- Boolean-based blind SQLi:
- Time-base: Tấn công dựa vào thời gian phản hồi từ app.

1.4.2.1. Boolean-based blind SQLi:

Đây là kiểu tấn công suy diễn dựa vào phản hồi dạng true,false từ web app.

Ví dụ: Giả sử trong một web, mỗi khi đăng nhập vào nó sẽ hiện thông báo “Welcome back”. URL có dạng như sau:



Hình 5: URL mẫu

Chú ý vào phần id = 2, ta sẽ dùng nó để khai thác thông tin.

Khi nhập vào web app như sau:

Username
2' and '1'='1'#

Hình 6: Đoạn mã nhập vào username

ID người dùng là 2, và '1' = '1' luôn đúng, do đó đã đăng nhập thành công, web hiện thông báo “Welcome back” -> Truy vấn True.

Ngược lại giả sử ta nhập vào ô input như sau:

Username
2' and '1'='2'#

Hình 7: Đoạn mã nhập vào username

Do '1' luôn khác '2' nên không thể đăng nhập được -> truy vấn False.

Từ đây, ta phát triển lên nhiều cách mới để có thể “mò” được mật khẩu chính xác của người dùng.

Username
2' and substring((select password from users where id = '2'),1,1)>'a'#

Hình 8: Đoạn mã nhập vào username

Câu này nghĩa là nếu kí tự đầu tiên trong password của người dùng có id 2 lớn hơn “a” (“b”, “c”, “d”...) thì sẽ trả về True (đăng nhập thành công) hoặc không lớn hơn “a” thì trả về false (không đăng nhập được).

Lại tiếp tục:

Username
2' and substring((select password from users where id = '2'),1,1)='b'#

Hình 9: Đoạn mã nhập vào username

Nếu câu này đăng nhập thành công, app hiện welcome back thì có nghĩa là ký tự đầu tiên trong mật khẩu của người này là chữ ‘b’. Mò dần dần, kết hợp với các câu lệnh kiểu khác (như tìm độ dài chuỗi) thì ta có thể biết được toàn bộ mật khẩu của người dùng này.

1.4.2.2. Time-base SQLi

Tương tự với kiểu boolean-base, time-base cũng là tấn công suy diễn dựa vào phản hồi từ máy chủ nhưng dựa vào ở dạng thời gian. Kỹ thuật buộc CSDL chờ một thời gian trước khi trả về kết quả. Thời gian trả về này sẽ cho phép kẻ tấn công biết được suy đoán của mình là đúng hay sai (thường là nếu trả về ngay lập tức thì truy vấn sai, còn trả về trễ một lúc thì là true).

1.4.3. Out of band SQLi

- -Cách này thường ít được dùng do nó cần sử dụng các tính năng mà phải được cho phép ở server và được dùng bởi app web.
- -Cách tấn công này không thể nhập đầu vào và thu về kết quả trên cùng một kênh (không dùng được In-band), hoặc khi server quá chậm hoặc không ổn định (không dùng được Inferential).

Kiểu tấn công này phụ thuộc vào khả năng server thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho kẻ tấn công.

1.5. *Cách phòng chống SQL injection*

1.5.1. “Làm sạch” hoàn toàn dữ liệu

Website cần có các bộ lọc để lọc toàn bộ input của người dùng. Ví dụ, tại ô “Nhập địa chỉ email” thì cần lọc “chỉ cho phép nhập các ký tự được phép xuất hiện trong email” (dấu @, chữ và số, dấu chấm, không có ký tự đặc biệt khác...) Cách thức này có thể bắt và chặn được đa số các nỗ lực đánh cắp dữ liệu thông qua kênh web.

Ví dụ, ta có thể lọc input đầu vào bằng một trong những cách tương đối hiệu quả bằng việc sử dụng regex. Giả sử đối với email, một email tiêu chuẩn sẽ có dạng

abc.123@abc.com bao gồm chữ hoặc số, đến dấu @, đến tên email cũng là chữ hoặc số, cuối cùng là tên miền. Như vậy regex cho email sẽ có dạng như sau:

The image shows a regex pattern: `/^[\w \.] + @ ([\w] + [\.]) + [\w] { 2 , 4 } $ /`. The pattern is color-coded: `^[\w \.]` is yellow, `@` is green, `([\w] + [\.])` is green, `+` is purple, `[\w]` is yellow, `{ 2 , 4 }` is blue, and `$` is purple.

Hình 10: Ví dụ về regex

Trong đó `\w` có nghĩa là cho phép chữ hoặc số, `\.` là cho phép dấu chấm; `+@([\w]+\.)` nghĩa là bắt buộc phải có @, theo sau @ là chữ hoặc số, sau đó đến dấu chấm; v.v...

1.5.2. Sử dụng ứng dụng tường lửa cho web (WAF - Web Application Firewall)

WAF là một thiết bị phần cứng hoặc phần mềm được cài vào máy chủ, thiết bị này có thể xử lý giao thức HTTP nhằm bảo vệ ứng dụng web. WAF kiểm tra lượng truy cập và sẽ lọc ra các yêu cầu có mối đe dọa xâm hại đến website trước khi đến ứng dụng web.

Các tường lửa cho app web có các bộ lọc phức tạp và thường xuyên được update cho các input tiềm ẩn nguy hiểm từ người dùng.

Một số tường lửa phổ biến: ModSecurity, AppTrana Managed Web Application Firewall,...

1.5.3. Tối thiểu hoá đặc quyền

Một chương trình yêu cầu thông tin đăng nhập của người dùng để chạy các lệnh SQL (chèn, cập nhật, tìm kiếm, xóa...).

Để giảm thiểu tác động trong một cuộc tấn công SQL injection, người dùng bình thường chỉ nên có một số đặc quyền cần thiết. Các đặc quyền bổ sung sẽ được cấp khi cần. Như vậy, thiệt hại của một cuộc tấn công SQL injection sẽ được giảm bớt đáng kể.

Ví dụ, một người dùng bình thường sẽ chỉ nên có các đặc quyền xem, sửa, xóa các dòng dữ liệu, và không được phép drop table (xóa bảng), alter table (sửa bảng),...

Privilege	Access
Read	Yes
Write	Yes
Update	Yes
Delete	Yes
DROP Table	None
Alter Table	None
Other Privileges	None

Hình 11: Mô tả đặc quyền cho người dùng bình thường

1.5.4. Tránh xây dựng câu lệnh truy vấn trực tiếp từ input của người dùng.

Kể cả có được lọc qua các bộ lọc như cách 1 thì input vẫn có nhiều lỗ hổng có thể tấn công.

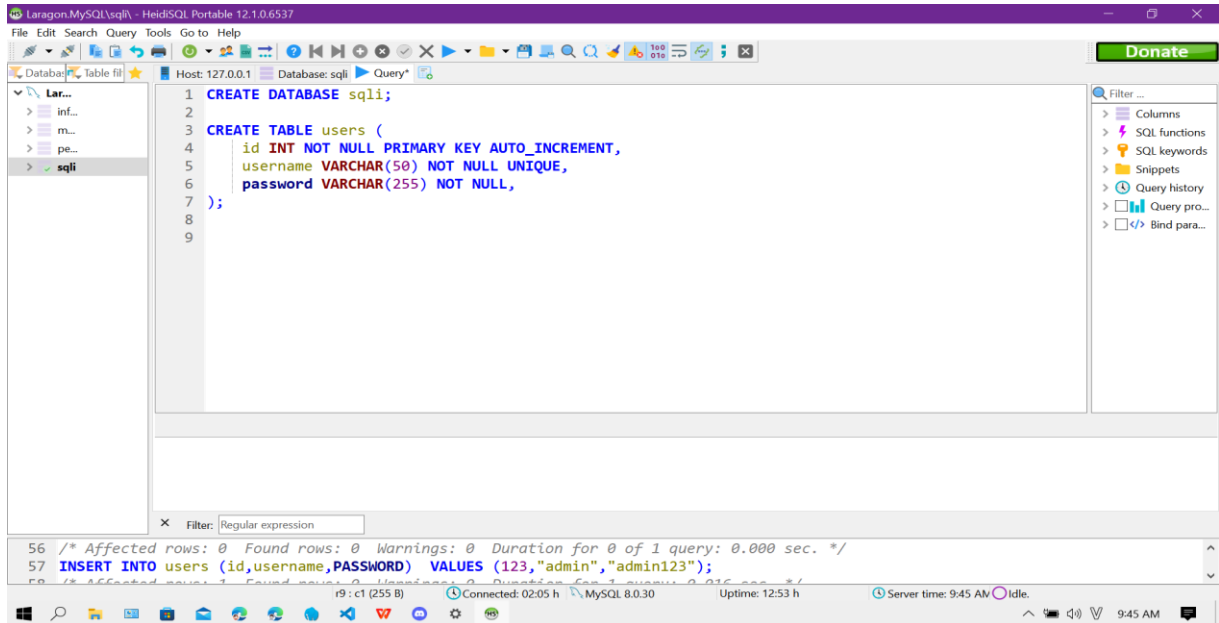
Do vậy, vẫn nên gán input vào các biến SQL và liên kết nó với câu lệnh hoặc dùng stored procedure, như vậy sẽ an toàn hơn nhiều.

2. Demo về SQL injection

2.1. Giải thích lý thuyết về code demo:

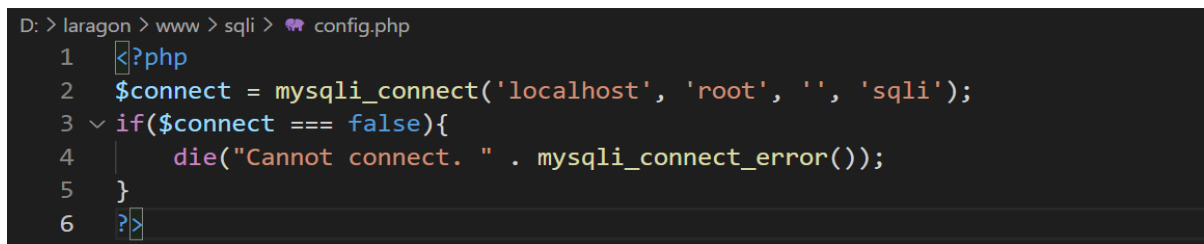
Ở trong phần này, em sử dụng PHP để tạo một web nhỏ có chức năng Đăng nhập, nếu đăng nhập thành công (username và password có trong database) thì chuyển hướng sang trang User, còn nếu không thì hiện dòng thông báo “Wrong password”; và dùng Laragon để tạo server cho web.

Đầu tiên, trong laragon, tạo một database đặt tên là “sql”. Trong sql tạo một bảng “users” để chứa dữ liệu username và password của người dùng.



Hình 12: Bảng “users” để chứa dữ liệu username và password

Tiếp theo, tạo file config, trong đó tạo biến connect dùng kết nối database với web.



Hình 13: File config

Kế đến là file login.

Nếu người dùng đã đăng nhập ngay trước đó rồi thì khi vào trang login sẽ chuyển hướng thẳng sang trang user.

```

<?php
session_start();

if (isset($_SESSION["logged"]) && $_SESSION["logged"] === true)
{
    header("location: user.php");
    exit;
}

```

Hình 14: File login

Phần xử lý input:

Nếu ô username hoặc password bị để trống thì hiện thông báo “Please enter username” hoặc “Please enter password”

```

9  require_once "config.php";
10 $username = "";
11 $password = "";
12 $username1 = "";
13 $password1 = "";
14 if ($_SERVER["REQUEST_METHOD"] == "POST")
15 {
16
17     /* Check if username is empty */
18     if (empty($_POST["username"]))
19     {
20         $username1 = "Please enter username.";
21     }
22     else
23     {
24         $username = $_POST["username"];
25     }
26
27     if (empty($_POST["password"]))
28     {
29         $password1 = "Please enter your password.";
30     }
31     else
32     {
33         $password = $_POST["password"];
34     }

```

Hình 15: Xử lý input

Nếu như cả 2 đều đã được điền đủ thì khởi tạo câu lệnh truy vấn để gửi tới database. Nếu như trong database có username và password này thì chuyển hướng sang trang user (đăng nhập thành công).

```

35     if (empty($username1) && empty($password1))
36     {
37         $sql = "SELECT * FROM users WHERE username = '$username' and password = '$password'";
38
39         $result = mysqli_query($connect, $sql);
40
41         if (mysqli_num_rows($result) > 0)
42         {
43             session_start();
44
45             $_SESSION["id"] = $id;
46             $_SESSION["logged"] = true;
47             $_SESSION["username"] = $username;
48
49             header("location: user.php");
50         }

```

Hình 16: Chuyển hướng đăng nhập

Dòng số 37 là dòng quan trọng, các dạng SQL injection hầu hết đều sẽ xoay quanh câu lệnh ở dòng 37. Khi đến phần “Cách phòng chống” chúng ta cũng sẽ làm việc trên phần code này.

Nếu không tìm thấy username và password trong database thì biến password1 được đặt là “sai”, hiện thông báo sai mật khẩu. Vẫn dừng ở trang login, đến khi login thành công (biến password1 và username1 rỗng) thì mới chuyển sang trang user.

Sau đó thì kết thúc kết nối tới database.

```

    else
    {
        $password1 = "Wrong password";
    }
    /* Close statement */
    mysqli_close($connect);
}
?>

```

Hình 17: Không tìm thấy user và pass thì kết nối tới database

Kế đến là đoạn code HTML để tạo hình cho trang login. Code này có sử dụng stylesheet từ thư viện bootstrap.css, được lưu trong thư mục stylehtml.

```

61 <!DOCTYPE html>
62 <html>
63 <head>
64     <meta charset="UTF-8">
65     <title>Login</title>
66     <link rel="stylesheet" href="stylehtml/bootstrap.css">
67     <style type="text/css">
68         body{ font: 14px ; }
69         .wrapper{ width: 400px; padding: 20px; }
70     </style>
71 </head>

72 <body>
73     <div class="wrapper">
74         <h2>Login</h2>
75         <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
76             <div class="form-group <?php echo (!empty($username1)) ? 'has-error' : ''; ?>">
77                 <label>Username </label>
78                 <input type="text" name="username" value="<?php echo $username; ?>">
79                 <span ><?php echo $username1; ?></span>
80             </div>
81             <div class="form-group <?php echo (!empty($password1)) ? 'has-error' : ''; ?>">
82                 <label>Password </label>
83                 <input type="password" name="password" >
84                 <span class="help-block"><?php echo $password1; ?></span>
85             </div>
86             <div class="form-group">
87                 <input type="submit" class="btn btn-primary" value="Sign in">
88                 Or <a href="signup.php"> Sign up</a></div>
89         </form>
90     </div>
91 </body>
92 </html>

```

Tiếp theo, em tạo file user.php, khi người dùng đăng nhập thành công sẽ chuyển hướng sang trang này. Nếu người dùng vào thẳng link /user.php mà chưa đăng nhập thì web tự động quay lại trang đăng nhập.

```

1 <?php
2 /* Initialize the session */
3 session_start();
4
5 /* Check if the user is logged in, if not then redirect him to login page */
6 if(!isset($_SESSION["logged"]) || $_SESSION["logged"] !== true){
7     header("location: login.php");
8     exit;
9 }
10 ?>

```

Code html:

```

12 <!DOCTYPE html>
13 <html>
14 <head>
15     <meta charset="UTF-8">
16     <title>Welcome</title>
17     <link rel="stylesheet" href="assets/bootstrap.css">
18 </head>
19 <body>
20     <div class="wrapper">
21         <h1>Hi, <b><?php echo htmlspecialchars($_SESSION["username"]); ?></b>. Welcome</h1>
22     </div>
23     <p>
24         <a href="logout.php" class="btn btn-primary">Sign Out</a>
25     </p>
26 </body>
27 </html>

```

Sau cùng là file logout.php để tạo lựa chọn đăng xuất sau khi đã đăng nhập thành công.

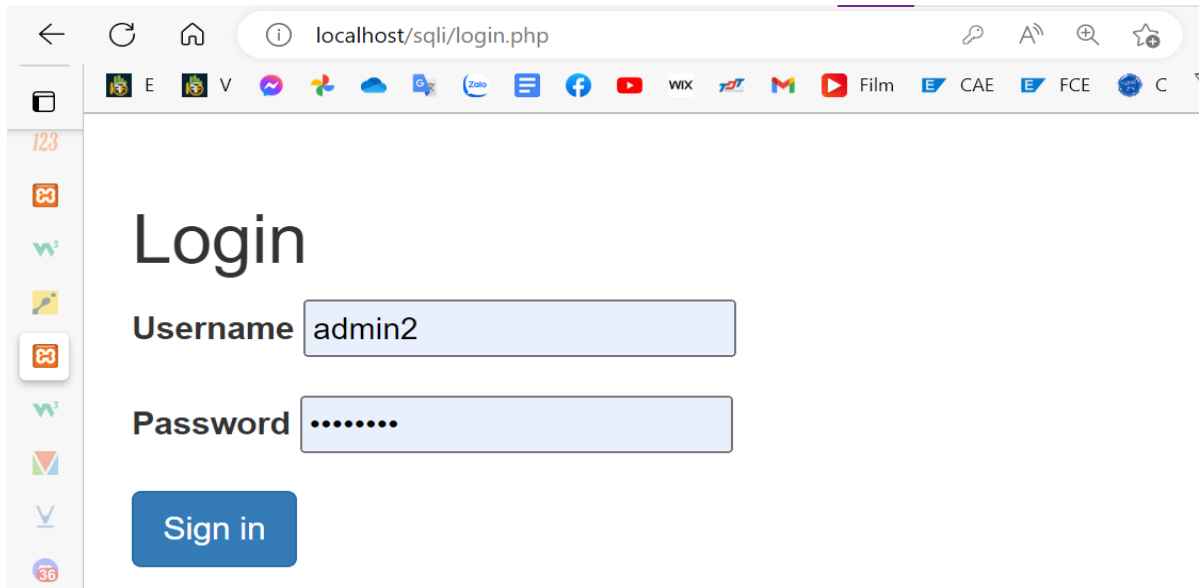
```

1 <?php
2 /* Initialize the session */
3 session_start();
4
5 /* Unset all of the session variables */
6 $_SESSION = array();
7
8 /* Destroy the session */
9 session_destroy();
10
11 /* Redirect to login page */
12 header("location: login.php");
13 exit;
14 ?>

```

Như vậy là về cơ bản đã xong một web đăng nhập đơn giản.

Ở trang đăng nhập có dạng như sau:



localhost/sqli/login.php

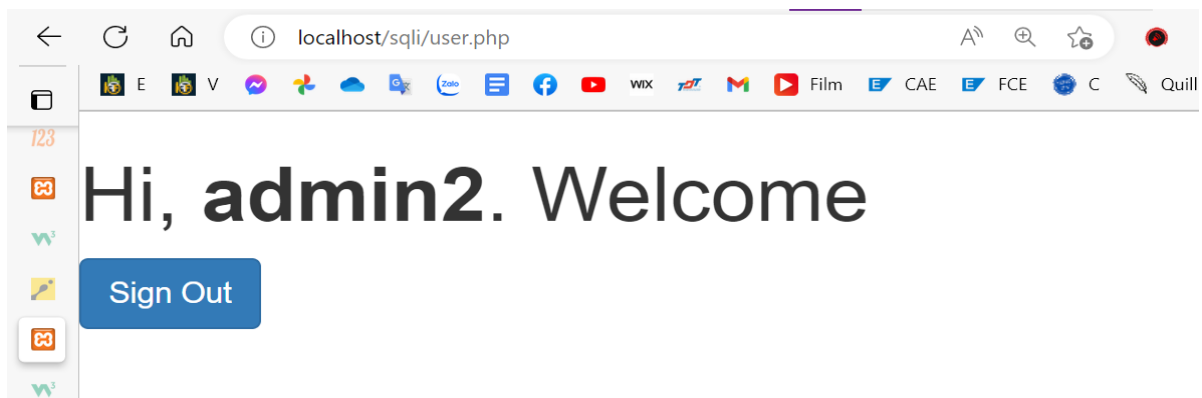
Login

Username

Password

[Sign in](#)

Và khi đăng nhập thành công thì chuyển đến trang user:



localhost/sqli/user.php

Hi, admin2. Welcome

[Sign Out](#)

2.2. Demo và chạy thử SQL injection:

Bây giờ ta bắt đầu thử injection. Trước tiên cần phải có dữ liệu trong database. Vào laragon để insert dữ liệu trong bảng users với tên người dùng là admin, password là admin123:

```
1 INSERT INTO users (id,username,PASSWORD)
2 VALUES (100,"admin","admin123")
```

Thử nhập sai input, kết quả thu được là thông báo wrong password:

Login

Username

Password

Wrong password

Sign in

Còn nếu nhập đúng thì chuyển hướng sang trang welcome như bên trên. Ta thử chèn mã với kiểu đơn giản nhất trước, Inband.

Login

Username

Password

Sign in

Web trả về thông báo đã đăng nhập thành công:

Hi, ' or 1=1#. Welcome

Log out

Hoặc trong trường hợp chưa biết database sử dụng SQL server hay Mysql nên không chắc về dấu comment (“#”) thì có thể chèn như sau:

Login

Username

Password

Sign in

Kết quả cũng là đăng nhập thành công tương tự như trên.

Hi, ' or '1'='1. Welcome

Log out

Thử với Error-base:

Login

Username

Password

[Sign in](#)

Kết quả sẽ là đoạn lỗi như hình

Fatal error: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'asd' at line 1 in D:\laragon\www\sql\login.php:39 Stack trace: #0 D:\laragon\www\sql\login.php(39): mysqli_query(Object(mysqli), 'SELECT * FROM u...') #1 {main} thrown in D:\laragon\www\sql\login.php on line 39

Trong trường hợp muốn biết cụ thể mật khẩu là gì, chứ không chỉ đơn thuần đăng nhập vào thì sao? Ta thử với Boolean-base SQLi.

Thử trên mật khẩu của người dùng admin chúng ta đã tạo ban đầu:

Nhập username như sau:

```
admin' and substring((select password from users where username = 'admin'),1,1)='a'##
```

Mật khẩu thì chọn đại mật khẩu gì cũng được vì đã comment nó rồi

Login

Username

Password

[Sign in](#)

Đăng nhập thành công, nghĩa là chữ cái đầu tiên trong mật khẩu của “admin” là “a”

Hi, admin' and substring((select password from users where username = 'admin'),1,1)='a'#. Welcome

Log out

Tiếp tục với các chữ cái thứ 2,3,... kết hợp với lấy ra độ dài mật khẩu, dần dần sẽ dò ra mật khẩu của admin.

Phương pháp phòng chống SQLi

***Cách 1: Dùng regex**

Vừa rồi chúng ta đã thử một vài kiểu SQL injection trên web đã tạo. Bây giờ, ta sẽ cải thiện web của mình để giảm thiểu khả năng bị tấn công bằng cách dùng *regex*.

Tạo một regex đơn giản “password và username chỉ cho phép chữ hoa, chữ thường hoặc số, không chứa các kí tự đặc biệt”:

```
$regex = '/[a-zA-Z0-9]/';
```

Nếu input của người dùng rỗng hoặc có bất cứ kí tự nào thoả mãn regex này, nghĩa là không phải chữ hoa, chữ thường, số, thì yêu cầu nhập lại. Nếu không có kí tự đặc biệt nào thì chấp nhận input. Sử dụng hàm `preg_match()` của Php để xác định xem input có thoả regex này không.

```
if (empty(trim($_POST["username"])) || preg_match($regex,$_POST["username"],$matches))
{
    $username1 = "Enter username.";
}
else
{
    $username = $_POST["username"];
}

if (empty(trim($_POST["password"])) || preg_match($regex,$_POST["password"],$matches))
{
    $password1 = "Enter password.";
}
```

Ta test thử lại trên web:

Khi nhập thử đoạn injection, có chứa các kí tự đặc biệt thì web bắt nhập lại:

Login

Username

Enter username.

Password

Sign in

*Cách 2: Giới hạn quyền cho người dùng trong database

Chỉ cấp các quyền như xem, sửa, xoá các dòng dữ liệu trong bảng cho người dùng bình thường, không cấp các quyền như alter table, delete table...

```
1 CREATE USER normal_user@localhost
2 IDENTIFIED BY '123';
3
4 GRANT INSERT, UPDATE, DELETE
5 ON sql_i.users
6 TO normal_user@localhost;
```

II. XSS INJECTION (TẤN CÔNG XSS)

1. Lý thuyết về tấn công XSS

1.1. Cross site scripting – XSS là gì?

Cross-Site Scripting (XSS) là một loại lỗ hổng bảo mật có thể được tìm thấy trong một số ứng dụng web. Nó là một kỹ thuật tấn công bằng cách chèn vào các website động (ASP, PHP, CGI, JSP ...) những thẻ HTML hay những đoạn mã SCRIPT nguy hiểm cho những nạn nhân sử dụng. Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng các Client-Site Script: JavaScript, Jscript, DHTML, HTML... Khi người dùng vào những trang web này thì mã độc sẽ được thực thi trên máy người dùng.

1.2. Mục đích thực hiện cuộc tấn công XSS

Tấn công XSS được thực hiện với nhiều mục đích như:

- Truy cập thông tin nhạy cảm hoặc bị hạn chế.
- Ăn cắp tiền (giao dịch ngân hàng, mua hàng online...)
- Theo dõi thói quen lướt web của người dùng.
- Thay đổi tính năng của trình duyệt.
- Bôi nhọ danh tiếng của một cá nhân hay công ty, tổ chức nào đó.
- Hủy hoại ứng dụng web,
- Tấn công từ chối dịch vụ...(cookie, keylogging, phishing)

1.3. Các bước thực hiện tấn công XSS

- Bước 1: Hacker tìm kiếm trang web chứa lỗ hổng XSS, chèn mã độc vào web đó.
- Bước 2: Hacker lừa người dùng vào trang web chứa mã độc bằng cách sử dụng kỹ thuật như phishing (*fishing for information* "câu thông tin" và *phreaking* "lừa đảo sử dụng điện thoại của người khác không trả phí").
- Bước 3: Sau khi người dùng nhấn vào đường link, đoạn mã độc hại được thực thi, các thông tin cần thiết như token, cookie..., sẽ được chuyển về máy chủ của hacker.
- Bước 4: Từ đó, hacker có thể thâm nhập vào tài khoản của người dùng, đăng nhập vào trang web mà không cần thông qua xác thực.

1.4. Các loại tấn công XSS

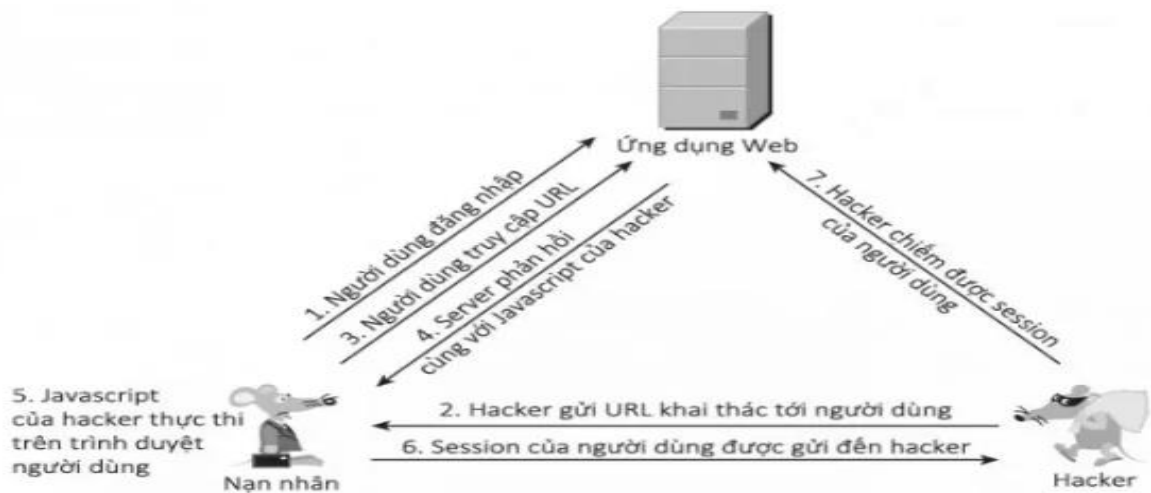
1.4.1. Reflected XSS (non-persistent XSS)

Loại tấn công này xảy ra khi một ứng dụng web phản hồi tới trình duyệt của user một tập lệnh độc hại. Mã độc thường được nằm ở trong URL, URL này sẽ gửi yêu cầu đến trang web có lỗ hổng. Hacker sẽ gửi trực tiếp URL có chứa mã độc cho user,

khi user nhấn vào URL này thì trang web sẽ được load chung với các đoạn script độc hại.

Reflected XSS là kiểu tấn công mà mã độc client gửi lên sẽ được thêm vào request và mã độc đó cũng sẽ được thêm vào trong response mà server trả về.

Reflected XSS xảy ra do đầu vào không được kiểm tra một cách kỹ càng dẫn đến việc kẻ tấn công có thể lợi dụng và chèn mã độc.

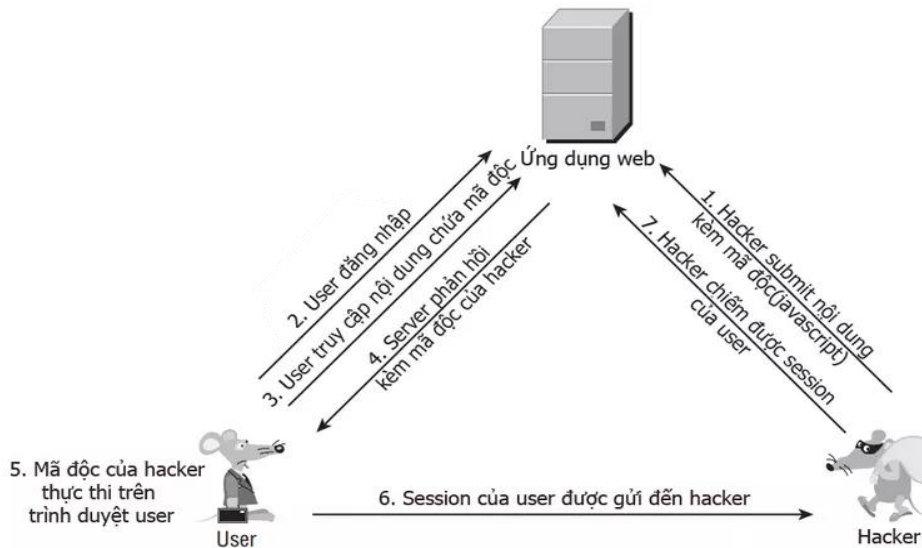


Hình 18: CÁC BƯỚC THỰC HIỆN TẤN CÔNG REFLECTED XSS

1.4.2. Stored XSS (persistent XSS)

Stored XSS là dạng tấn công mà hacker chèn các mã độc vào server điển hình là cơ sở dữ liệu. Dạng tấn công này xảy ra khi các dữ liệu được gửi lên server không được kiểm tra kỹ lưỡng mà lưu trực tiếp vào server. Khi người dùng truy cập vào trang web này thì những đoạn script độc hại được tải từ server sẽ được thực thi chung với quá trình tải trang web. Các đoạn script thường được chèn vào website thông qua các chức năng như: viết bình luận, đăng bài, đơn góp ý, ...

Vì vậy, Stored XSS chính là kiểu tấn công nguy hiểm nhất vì mã độc được lưu trên server và sẽ được thực thi trên tất cả người dùng truy cập đến nó.



Hình 19: CÁC BƯỚC THỰC HIỆN TẤN CÔNG STORED XSS

1.4.3. DOM-based vulnerabilities

DOM viết tắt của Document Object Model là 1 tiêu chuẩn của W3C đưa ra nhằm để truy xuất và thao tác dữ liệu của tài liệu có cấu trúc như HTML, XML. Mô hình này thể hiện tài liệu dưới dạng cấu trúc cây phân cấp. Mỗi thành phần trong HTML, XML được xem như một node.

HTML DOM là 1 mô hình tiêu chuẩn cho các tài liệu HTML. HTML DOM sẽ xác định những thứ sau:

- Các phần tử HTML (các thẻ) như các đối tượng. Các phần tử này được biểu diễn dưới dạng cây phân cấp.
- Các tính chất của tất cả các phần tử HTML (innerHTML, innerText...).
- Các sự kiện của tất cả các phần tử html (click, focus...).
- Các phương thức để thao tác vào các phần tử html.

Để hiểu thêm về DOM-based XSS ta cần biết về 2 khái niệm:

- **Source:** là những vị trí mà mã độc sẽ chứa ở trong đó và ứng dụng sẽ lấy mã độc ra từ vị trí đó. Có 4 loại source khác nhau:

URL-BASED SOURCES	NAVIGATION-BASED SOURCES	COMMUNICATION SOURCES	STORAGE SOURCES
location	window.name	Ajax	Cookies
location.href	document.referrer	Web Sockets	localStorage
location.search		Window Messaging	SessionStorage
location.pathname			

Hình 20: Các loại source và các kiểu thể hiện

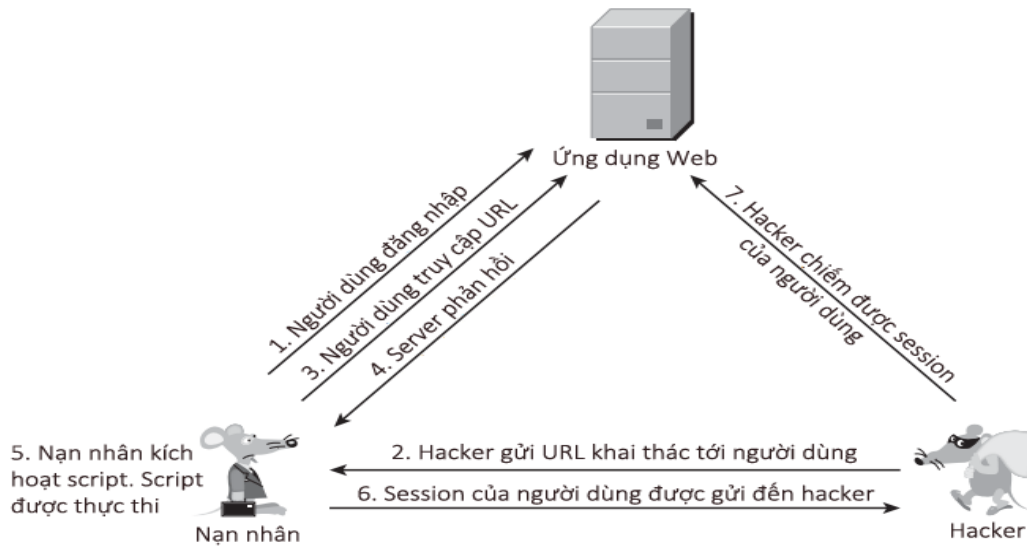
- **Sink:** là nơi mà mã độc được truyền vào từ source và được thực thi. Nói cách khác, sink là các method hoặc property nguy hiểm có thể thực hiện mã độc được lấy từ source. Có 3 loại sink khác nhau:

JAVASCRIPT EXECUTION SINKS	HTML EXECUTION SINKS	JAVASCRIPT URI SINKS
eval()	innerHTML()	location
setTimeout()	outerHTML()	location.href
setInterval()	document.write()	location.replace()
Function()		location.assign()

Hình 21: Các loại sink và các kiểu thể hiện

DOM-based vulnerabilities là gì?

DOM-based XSS là dạng tấn công khai thác quá trình xử lý dữ liệu được thực hiện bên phía máy khách (client) thường dùng JavaScript để xử lý. Mã độc sẽ được đọc ra từ các source và được đưa vào các sink sau đó mã độc sẽ được thực thi. Việc đọc mã độc và thực thi mã độc sẽ diễn ra hoàn toàn ở trình duyệt.



Hình 22: CÁC BƯỚC THỰC HIỆN TẤN CÔNG DOM-BASED XSS

1.5. Các biện pháp phòng chống tấn công XSS

1.5.1. Lọc (XSS Filter)

- Không tin bất kỳ dữ liệu nào do người dùng đưa vào. Lọc các thẻ và các thuộc tính có thể hỗ trợ việc tấn công bằng xss (<script>, , onerror, onfocus...).
- Sử dụng các bộ lọc tự tạo hoặc từ thư viện để lọc bỏ các thẻ HTML/CSS/scripts khỏi dữ liệu nhập từ người dùng. Sử dụng các thư viện như Html Sanitizer có sẵn của OWASP để làm sạch dữ liệu đầu vào bằng cách tạo ra một danh sách các thẻ và thuộc tính được phép sử dụng. Ngoài ra còn có DOMPurity, PHP HTML Purifier, Python Bleach,...
- Sử dụng biểu thức chính quy (Regular Expressions) để tăng hiệu quả lọc
- Các bộ lọc cần được cập nhật thường xuyên để có thể theo kịp sự thay đổi của các XSS

- Các bộ lọc dữ liệu nhập phải được thực hiện trên máy chủ (server-sidescripts) do các bộ lọc trên máy khách có thể bị loại bỏ một cách dễ dàng.

=> Có thể gây khó khăn cho người dùng trong việc nhập text.

1.5.2. Thoát khỏi XSS (XSS Escape)

Vô hiệu hóa tấn công XSS bằng cách thay thế các ký tự riêng của HTML/scripts để chuyển các đoạn mã có thể thực hiện thành dữ liệu thông thường có thể thực hiện được. XSS Escape có thể chặn XSS mà người dùng không bị hạn chế khi nhập text.

Kí tự	Chuyển đổi
&	&
<	<
>	>
"	"
'	'

Hình 23: Ví dụ về 5 ký tự được chuyển đổi của HTML

1.5.3. Đặt cờ HttpOnly

HttpOnly là cờ bổ sung được thêm vào trong HTTP response header Set-Cookie. Mục đích của thuộc tính httponly là bảo vệ cookie khỏi việc truy cập trái phép từ browser. Chỉ lưu và gửi kèm cookie phản hồi từ client tới server. Việc này làm hạn chế sự can

thiệt từ trình duyệt giúp hạn chế rủi ro từ các cuộc tấn công đánh cắp cookie. Nếu cookie được set cờ HttpOnly, nó không thể bị truy cập bởi mã Javascript.

⇒ Điều đó có nghĩa hacker sẽ không thể nhận được cookie.

1.5.4. Sử dụng CSP

CSP là chính sách bảo mật nội dung, được sử dụng để xác định các nguồn nội dung an toàn trên website mà trình duyệt có thể tải về cho người dùng. CSP là biện pháp đối phó rất hiệu quả với kiểu hack chèn mã độc Cross Site Scripting (XSS).

Để cho phép CSP, chúng ta cấu hình cho server trả về Content-Security-Policy HTTP header hoặc có thể sử dụng thông qua thẻ <meta>.

Ví dụ:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; img-src https://*; child-src 'none';">
```

Tên directive:

- script-src: chỉ định nguồn (nơi) load các tài nguyên js
- style-src: chỉ định nguồn (nơi) load các tài nguyên css
- image-src: chỉ định nguồn (nơi) load các tài nguyên img
- font-src: chỉ định nguồn (nơi) load các tài nguyên font
- frame-src: chỉ định nguồn (nơi) load các tài nguyên frame

Các giá trị của CSP directive:

- *: là wildcard, tất cả
- self: chỉ định domain đang truy cập
- none: không cho phép bất kỳ nguồn nào
- http://www.domain.com: cho phép tải resource từ domain được chỉ định, cái này khác hoàn toàn với domain.com
- domain.com: cho phép tải resource từ domain chỉ định, không cho phép từ http://www.domain.com, subdomain.domain.com, cdn.domain.com, ...

- *.domain.com: cho phép load resource từ subdomain của domain domain.com
- https: cho phép load từ những trang https

Ví dụ 1: Content-Security-Policy: default-src 'self'

Chỉ cho phép tải resource từ domain hiện tại hoặc domain đang truy cập

Ví dụ 2:

Content-Security-Policy: default-src 'self' *.trusted.com

Cho phép tải resource từ domain hiện tại và từ các subdomain khác của domain trusted.com

2. Phần mềm cũng như cài đặt hệ thống demo XSS

Ứng dụng web dễ bị tổn thương (DVWA) là một ứng dụng web PHP/MySQL rất dễ bị tấn công. Mục tiêu chính của nó là hỗ trợ các chuyên gia bảo mật kiểm tra kỹ năng và công cụ của họ trong môi trường hợp pháp, giúp các nhà phát triển web hiểu rõ hơn về các quy trình bảo mật ứng dụng web và hỗ trợ cả sinh viên và giáo viên tìm hiểu về bảo mật ứng dụng web trong môi trường được kiểm soát. Môi trường lớp học.

Mục đích của DVWA là thực hành một số lỗ hổng web phổ biến nhất, với nhiều mức độ khó khác nhau, với giao diện đơn giản dễ hiểu. Xin lưu ý, có cả lỗ hổng được ghi lại và không có giấy tờ với phần mềm này. Đây là cố ý. Bạn được khuyến khích thử và khám phá càng nhiều vấn đề càng tốt.

Bước 1: Tải xuống DVWA:

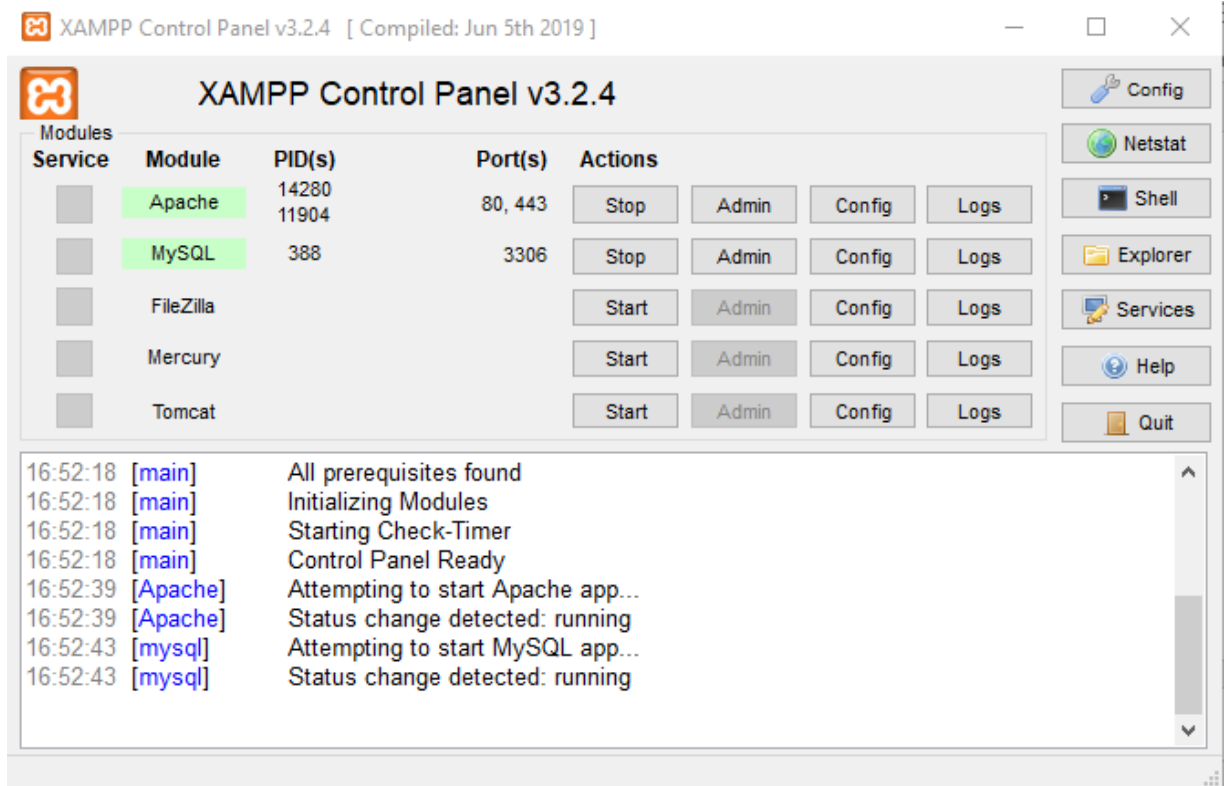
<https://sourceforge.net/projects/dvwa.mirror/>

Tải xuống và cài đặt XAMPP trên máy tính của bạn. XAMPP là gói giải pháp máy chủ web đa nền tảng mã nguồn mở và miễn phí được phát triển bởi Apache Friends, bao gồm chủ yếu là Máy chủ HTTP Apache, cơ sở dữ liệu MariaDB và trình thông dịch cho các tập lệnh được viết bằng ngôn ngữ lập trình PHP và Perl

Bước 2: Liên kết tải xuống XAMPP:

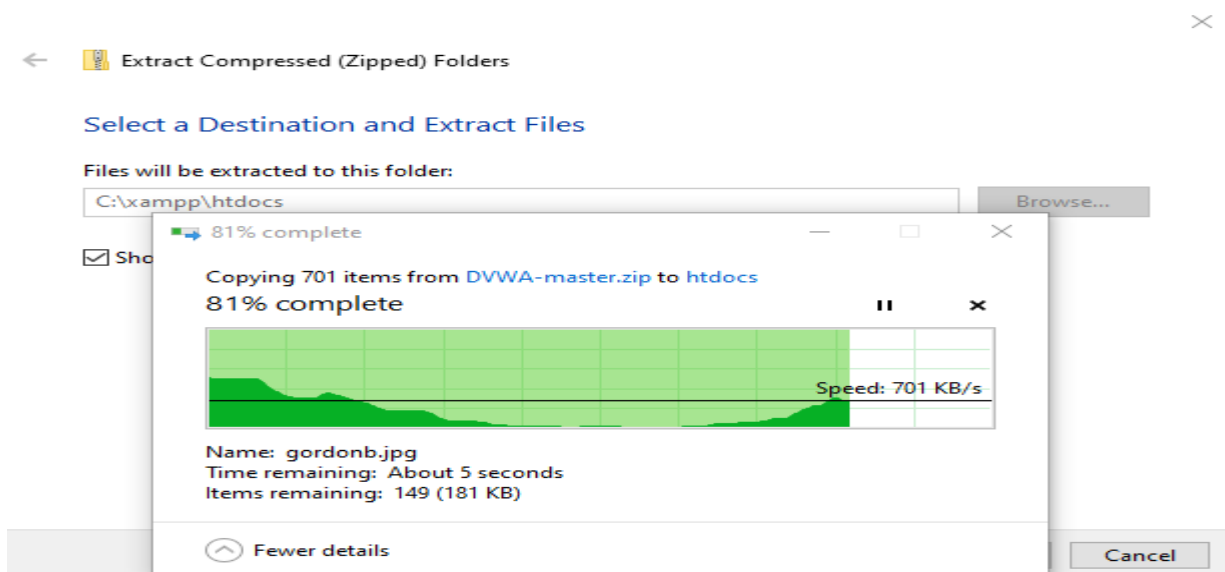
<https://www.apachefriends.org/download.html>

Bước 3: Mở XAMPP và bắt đầu 'Apache và MySQL'



Hình 24: Mở Xampp và bấm Start với 2 mục đầu

Bước 4: Giải nén tệp DVWA trong htdocs sẽ có sẵn trong C:\xampp




Hình 25: Giải nén tệp htdocs từ file DVWA đã tải

Bước 5: Mở thư mục htdocs và đổi tên 'DVWA-master' thành 'dvwa'

Bước 6: Và Mở trình duyệt của bạn rồi gõ 127.0.0.1/dvwa

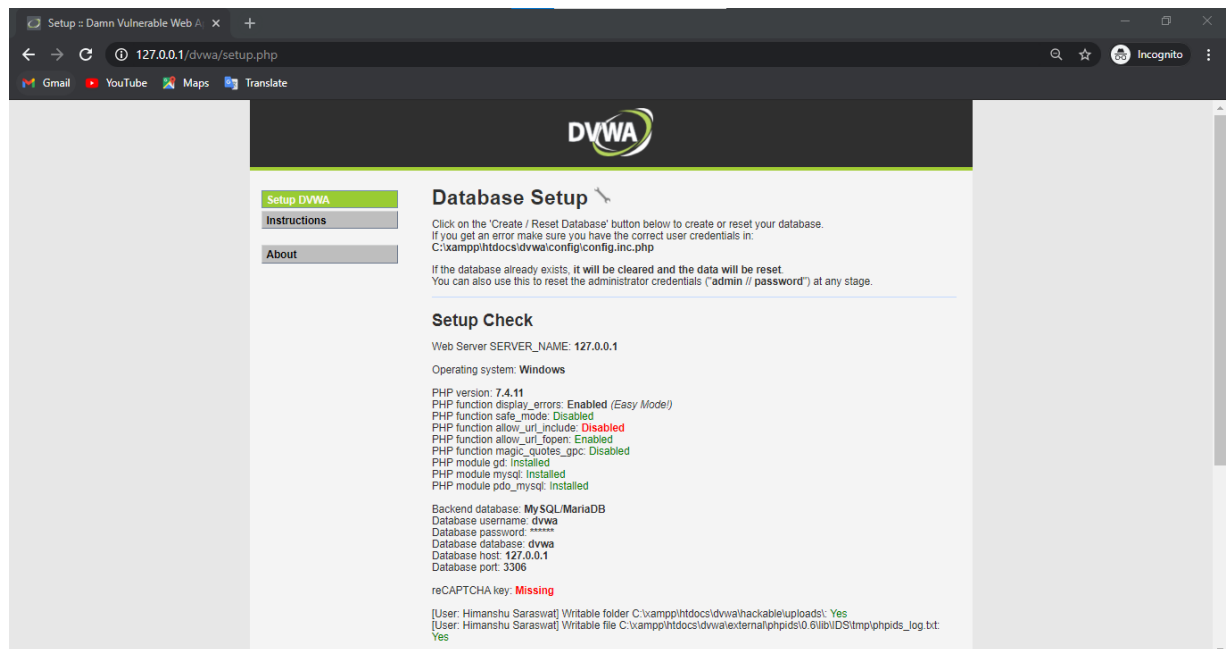
Nó sẽ hiển thị loại lỗi này “Lỗi hệ thống DVWA – không tìm thấy tệp cấu hình. Sao chép config/config.inc.php.dist sang config/config.inc.php và định cấu hình cho môi trường của bạn.”

Bước 7: Tên tệp 'config.inc.php.dist' đổi tên nó thành 'config.inc.php', nó sẽ có sẵn trong C:\xampp\htdocs\dvwa\config

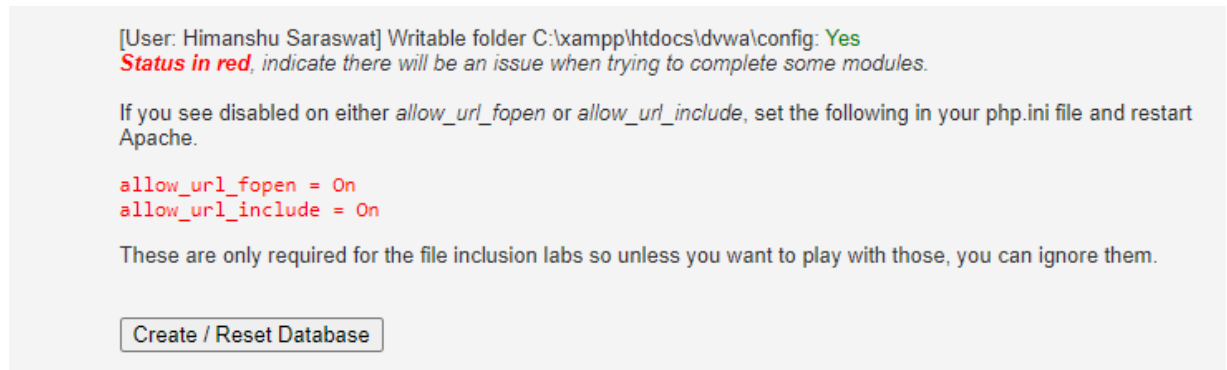
Name	Date modified	Type	Size
 config.inc.php	11-11-2020 17:07	PHP File	2 KB

Hình 26: Đổi tên file và thể loại đuôi file config

Bước 8: Sau đó gõ lại trình duyệt với localhost 1 lần nữa



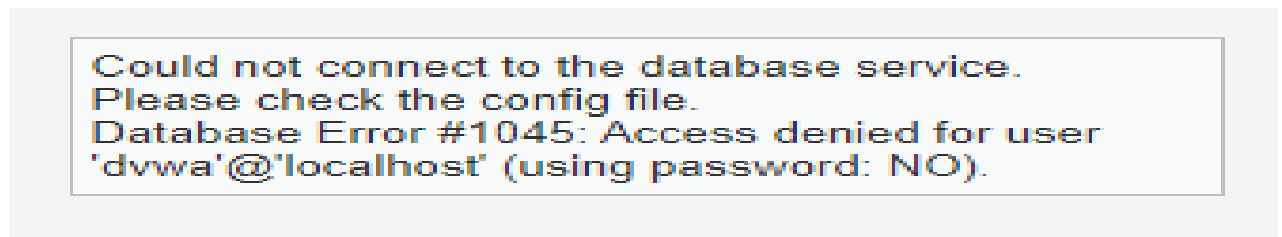
Bước 8: Nhấn Create/Reset Database



Hình 27: Nhấn Create/Reset Database

Bước 9: Loại lỗi dưới đây sẽ xuất hiện và bạn mở notepad thực hiện sửa giống ảnh số 12

Bây giờ, bạn cần chỉnh sửa tệp cấu hình mà bạn đã đổi tên ở bước trước đó, Sau đó, mở nó trong Notepad và trong tab mật khẩu, xóa mật khẩu hoặc làm trống mật khẩu bằng cách xóa mật khẩu mặc định và đặt tên người dùng là root. (its user database user id, password)



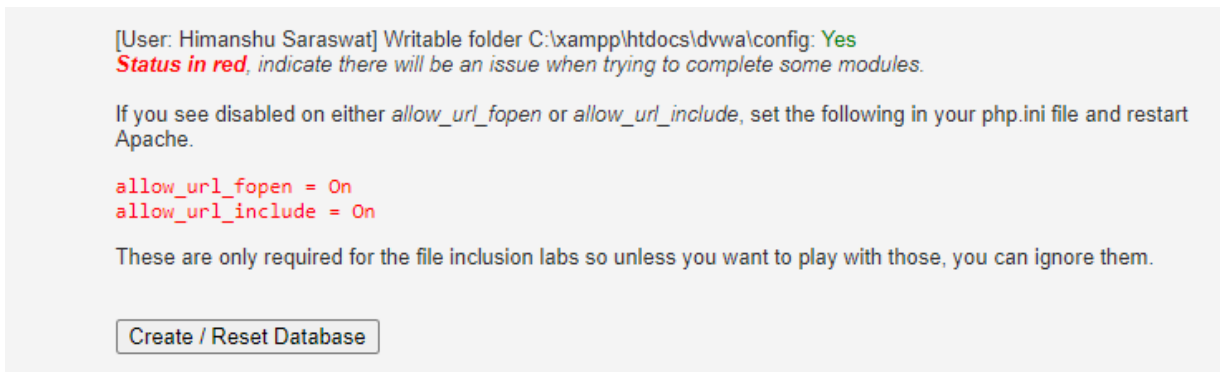
Hình 28: Lỗi xuất hiện


```
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ] = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = '';
$_DVWA[ 'db_port' ] = '3306';

# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ] = '';
$_DVWA[ 'recaptcha_private_key' ] = '';
```

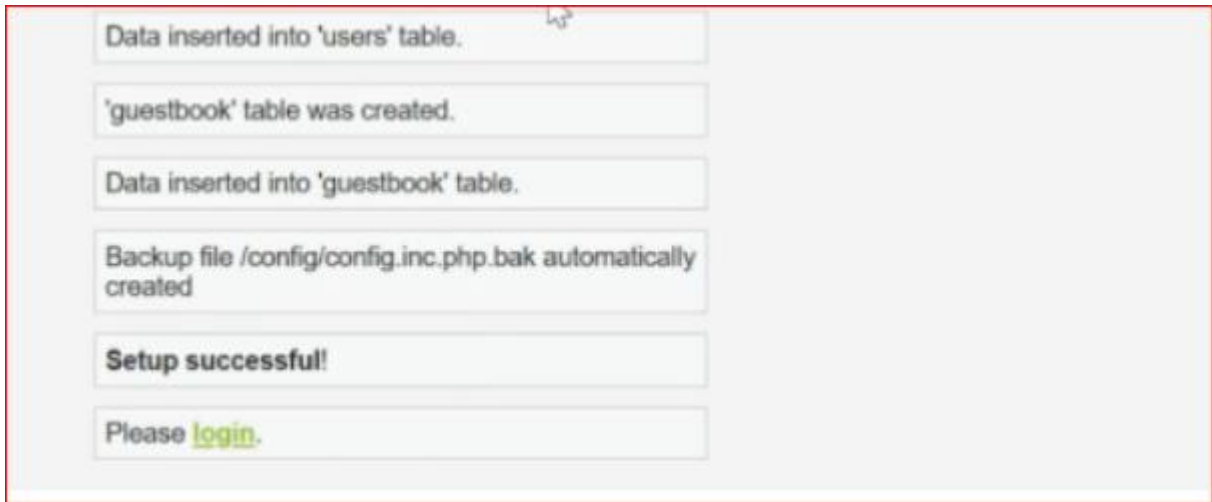
Hình 29: Chỉnh sửa lại 2 dòng trong file Notepad

Bước 10: Tiếp theo nhấn lại Create/Reset Database



Hình 30: Nhấn lại Create/Reset Database

Bước 11: Xuất hiện nút login và nó sẽ tự động chuyển hướng đến trang đăng nhập.

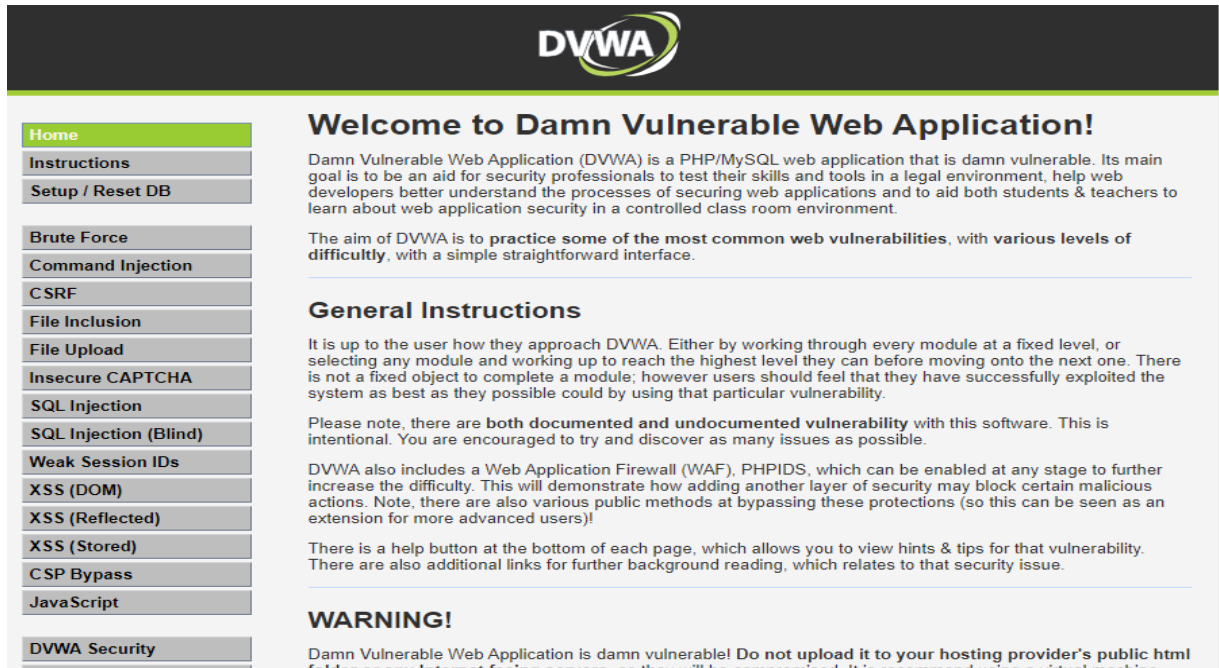


Hình 31: Nút login ở cuối

Bước 12: Sau khi giao diện đăng nhập xuất hiện bạn tiến hành nhập thông tin:
Tên người dùng mặc định là 'admin' và mật khẩu là 'password'.

A screenshot of the DVWA login form. At the top is the DVWA logo, which consists of the letters 'DVWA' in a bold, dark blue font, with a green and blue swoosh graphic to the right. Below the logo are two input fields: 'Username' and 'Password'. The 'Username' field is a single-line text input. The 'Password' field is a single-line text input. Below these fields is a 'Login' button with a blue border and the word 'Login' in blue text.

Hình 32: Hình ảnh form login hiện lên



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

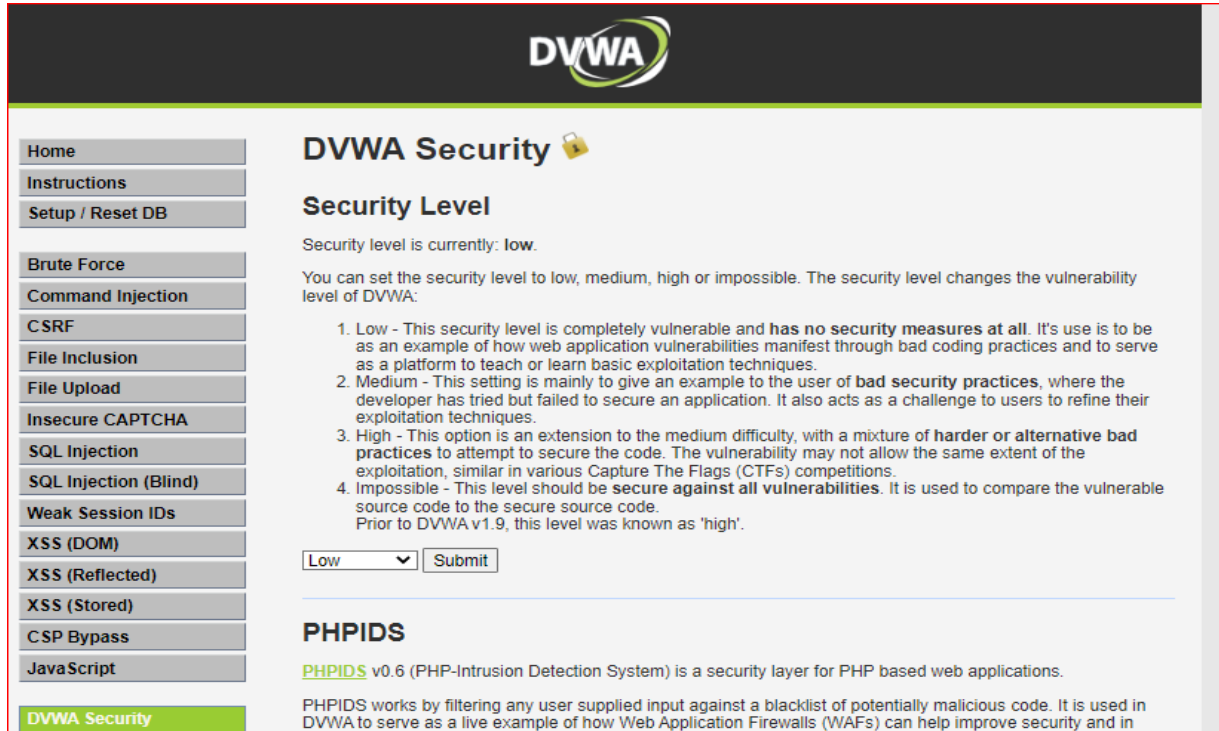
There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! Do not upload it to your hosting provider's public html folder or you Internet facing server, as they will be compromised. It is recommended using a virtual machine.

Hình 33: Giao diện bạn đã tải xong phần mềm

3. Demo các loại tấn công XS



DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Low

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in

Hình 34: Thiết lập DVWA Security theo các mức độ

Trước khi tiến hành các loại demo chúng ta sẽ phải thiết lập mức độ bảo mật, độ an toàn của localhost khi test các loại tấn công. Có 4 mức độ test cho mỗi demo:

- Low: mức bảo mật này hoàn toàn dễ bị tổn thương và không có biện pháp bảo mật nào cả. Nó được sử dụng để làm ví dụ về cách các lỗ hổng ứng dụng web biểu hiện thông qua các hoạt động mã hóa không tốt và dùng làm nền tảng để dạy hoặc học các kỹ thuật khai thác cơ bản.
- Medium: cài đặt này chủ yếu là để đưa ra ví dụ cho người dùng về các phương pháp bảo mật kém, trong đó nhà phát triển đã cố gắng bảo mật ứng dụng nhưng không thành công. Nó cũng hoạt động như một thách thức đối với người dùng để tinh chỉnh các kỹ thuật khai thác của họ.
- High: tùy chọn này là một phần mở rộng cho độ khó medium, với sự kết hợp của các phương pháp xấu hơn hoặc thay thế để cố gắng bảo mật mã.
- Impossible: mức này phải được bảo mật trước mọi lỗ hổng. Nó được sử dụng để so sánh mã nguồn dễ bị tổn thương với mã nguồn an toàn.

3.1. Demo Reflected XSS

Dưới đây là demo đúng về thông tin lúc ban đầu của hệ thống và không có lỗi xảy ra.

192.168.18.209/dvwa/vulnerabilities/xss_r/

Tạo tin

DVWA

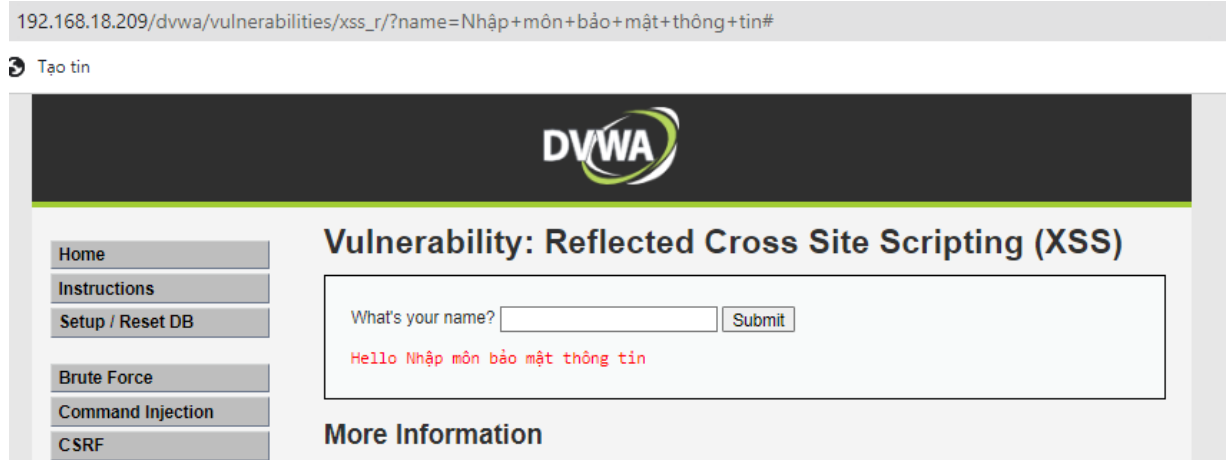
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

More Information

Hình 35 Demo nhập môn bảo mật thông tin



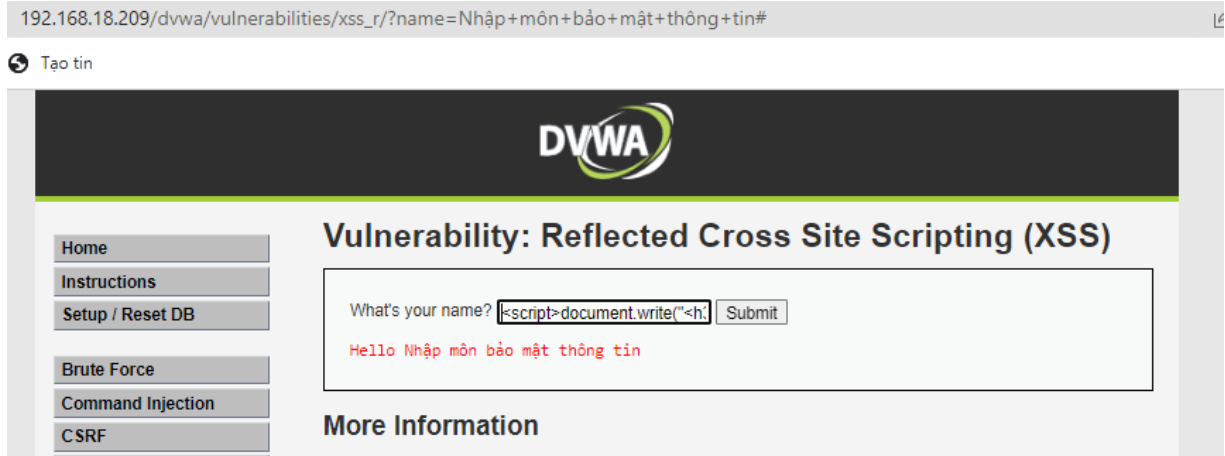
Hình 36: Hiển thị kết quả của demo

3.1.1. Mức độ an toàn thấp – Security Low

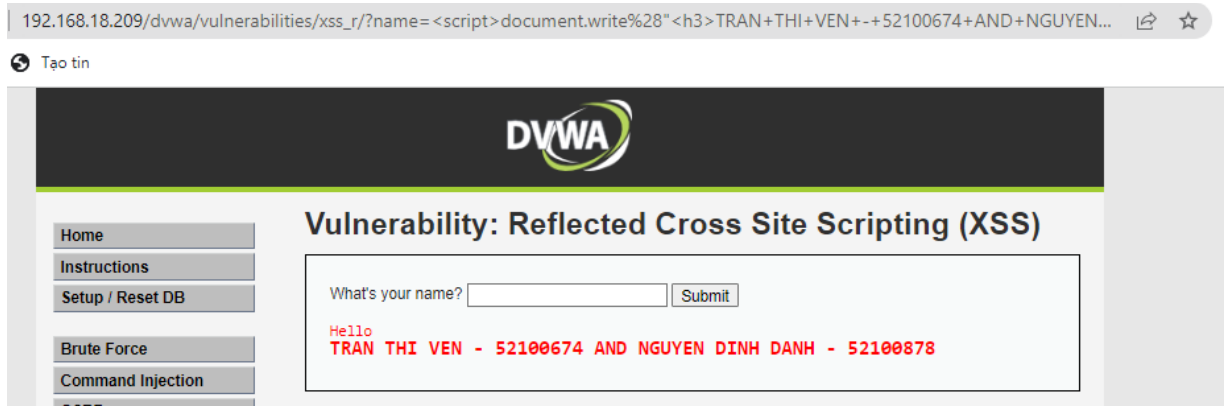


Hình 37: Source code ở mức độ thấp

Chúng ta sẽ thực hiện hiện mã hóa thông tin bằng cách tấn công input nhập tên của hệ thống. Do đoạn code không có mã hóa đoạn scrip nên tận dụng sơ hở này nhóm em chèn vào một đoạn script để thấy được lỗi của hệ thống. Chèn đoạn script để tấn công vào ô input: **<script>document.write("<h3>TRAN THI VEN - 52100674 AND NGUYEN DINH DANH - 52100878</h3>");</script>**



Hình 38: Chèn script vào ô input



Hình 39: Kết quả sau khi chèn

Sau khi chèn người dung nhận được câu lệnh chào to hơn bình thường do chúng em đã dùng thẻ <h3> để định dạng câu chào.

3.1.2. Mức độ an toàn trung bình – Security Medium

Reflected XSS Source

vulnerabilities/xss_r/source/medium.php

```
<?php
header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

Hình 40: Source code ở mức độ trung bình

Khi nhìn source ở mức độ trung bình đã bị mã hóa và thay thế chuỗi script thành tên nên chúng ta không thể nào lấy dữ liệu được.

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello alert(document.cookie)

Hình 41: Khi dùng script mã hóa không được

Nhưng đối với source code này ta thấy được nó chỉ mã hóa đối với từ script thường nên ta có thể chèn được 1 script nào bất kì với 1 hoặc nhiều kí tự viết hoa.

Script: `<sCriPt>alert(document.cookie)</sCriPt>`

[Home](#)
[Instructions](#)
[Setup / Reset DB](#)
[Brute Force](#)
[Command Injection](#)
[CSRF](#)

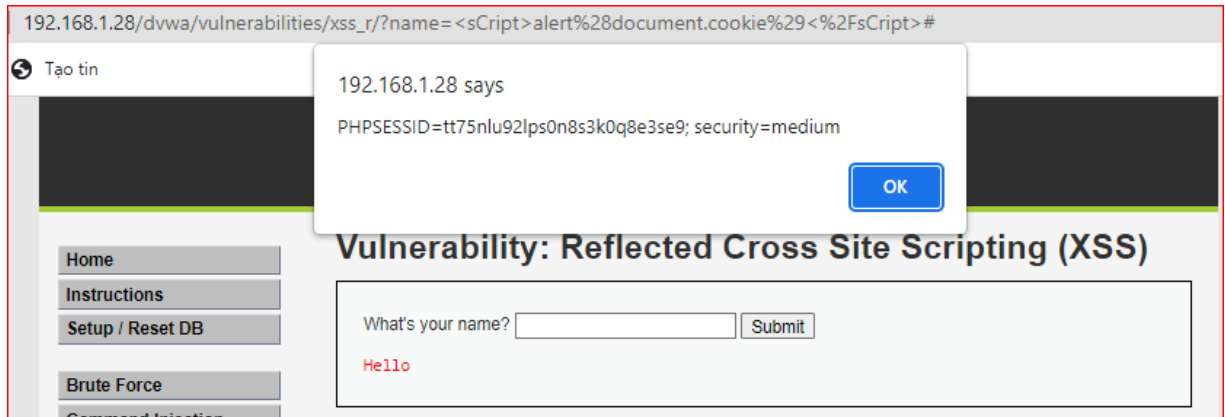
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello |

More Information

Hình 42: Insert đoạn script ở dạng khác

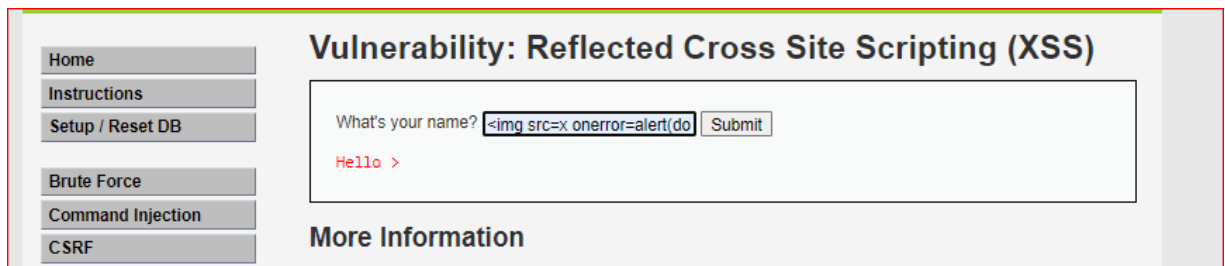


Hình 43: Kết quả vẫn bị ăn cắp cookie

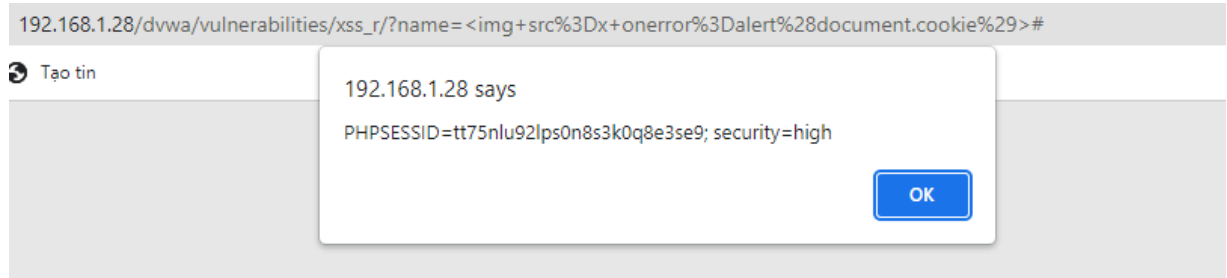
3.1.3. Mức độ an toàn cao – Security High



Hình 44: Source code ở mức độ an toàn cao

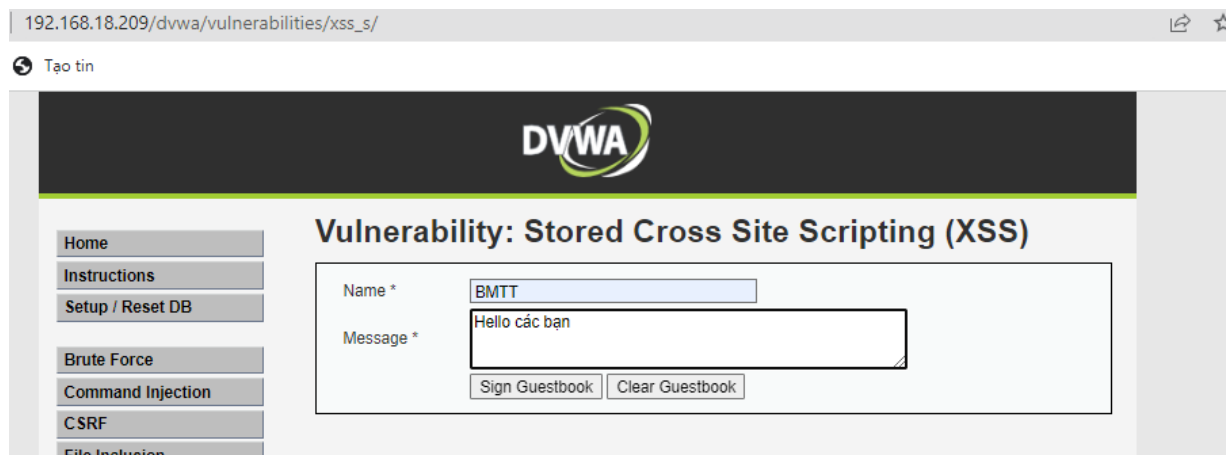


Hình 45: Chèn thẻ img ở high level

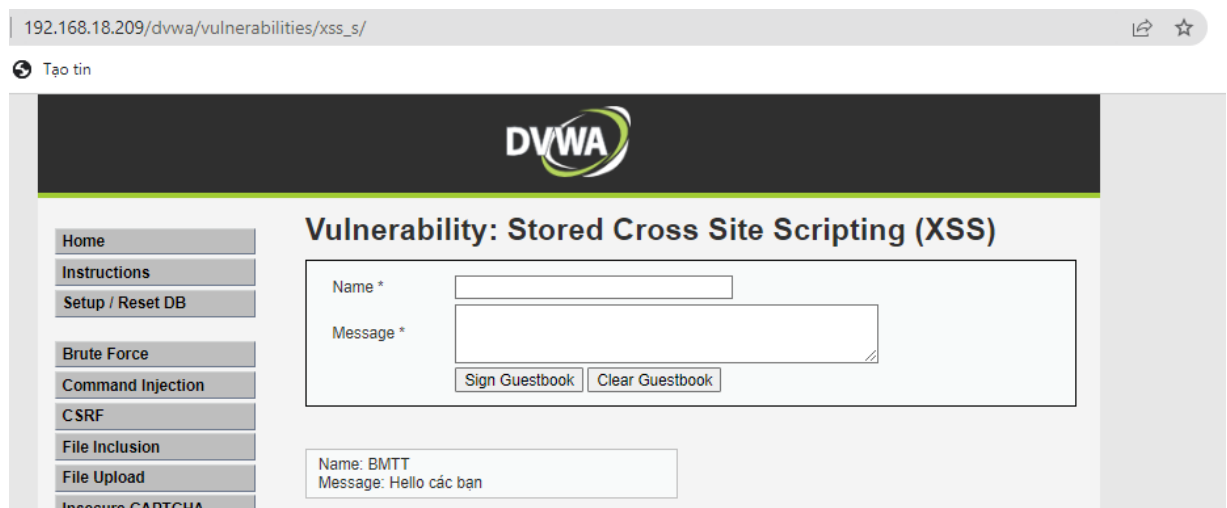


Hình 46: Thành công lấy được cookie

3.2. Demo Stored XSS



Hình 47: Nhập demo đúng



Hình 48: Demo đúng và xuất hiện thông điệp

3.2.1. Mức độ an toàn thấp – Security Low

```

vulnerabilities/xss_s/source/low.php

<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = stripslashes( $message );
    $message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

    // Sanitize name input
    $name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

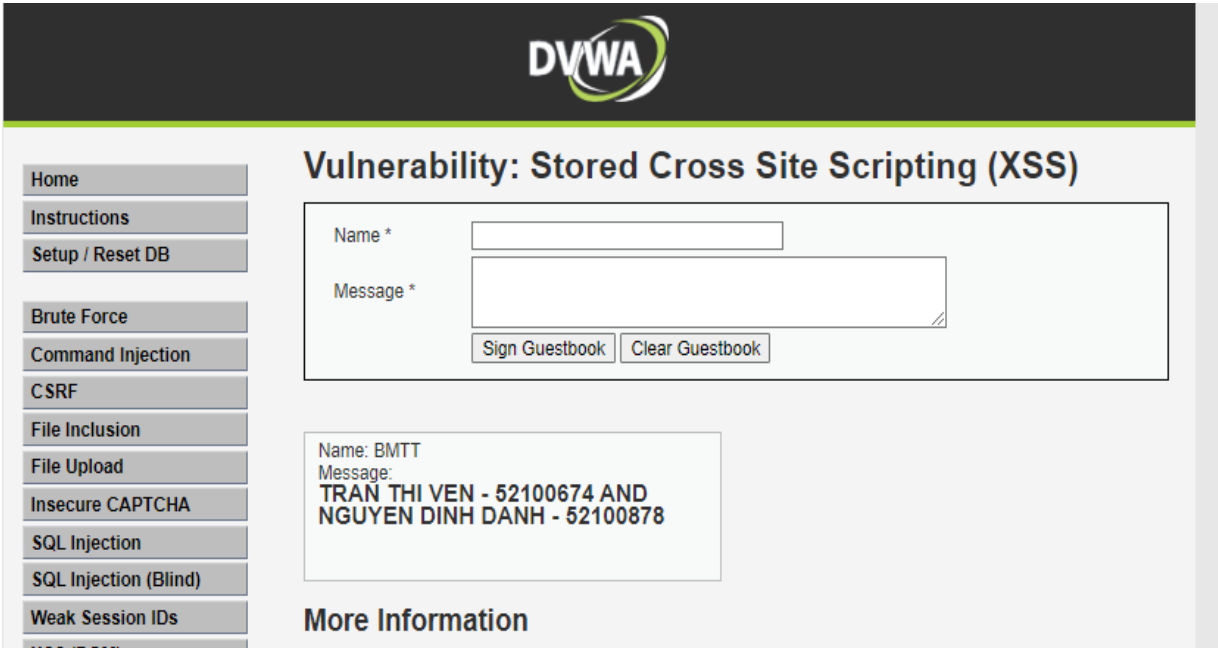
    // Update database
    $query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '

```

Hình 49: Source code của mức độ an toàn thấp

Nhận ra được lỗ hổng của source, nên chúng em tiếp tục chèn thẻ script chứa `<h3>` vào để quan sát:

Hình 50: Chèn thẻ script vào để demo stored xss



Hình 51: Thẻ `<h3>` đã làm thay đổi cỡ chữ của localhost này

3.2.2. Mức độ an toàn trung bình và cao – Security Medium and High

```
vulnerabilities/xss_s/source/high.php

<?php

if( isset( $_POST[ 'btnSign' ] ) ){
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"],  $message ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) > "" ? ""));
    $message = htmlentities( $message );

    // Sanitize name input
    $name = preg_replace( '/<(.*?)>s(<.*>)(.)r(<.*>)j(<.*>t/i','', $name );
    $name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"],  $name ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) > "" ? ""));

    // Update database
    $query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["__mysqli_ston"],  $query ) or die( '

```
'. (($is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error())
)) . '
```

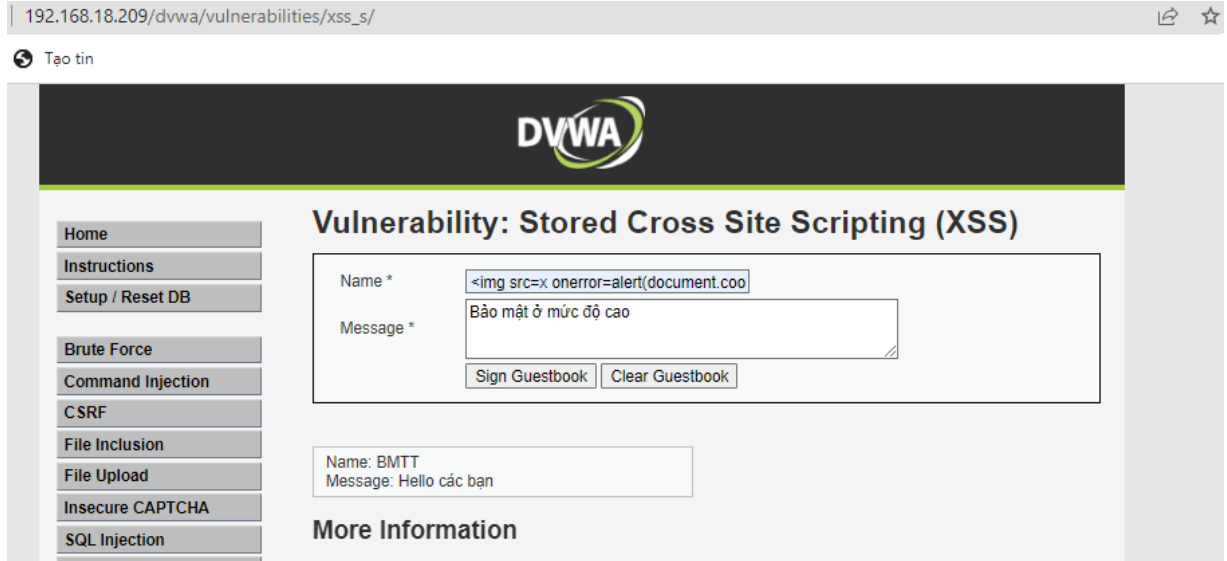
' );

    //mysql_close();
}

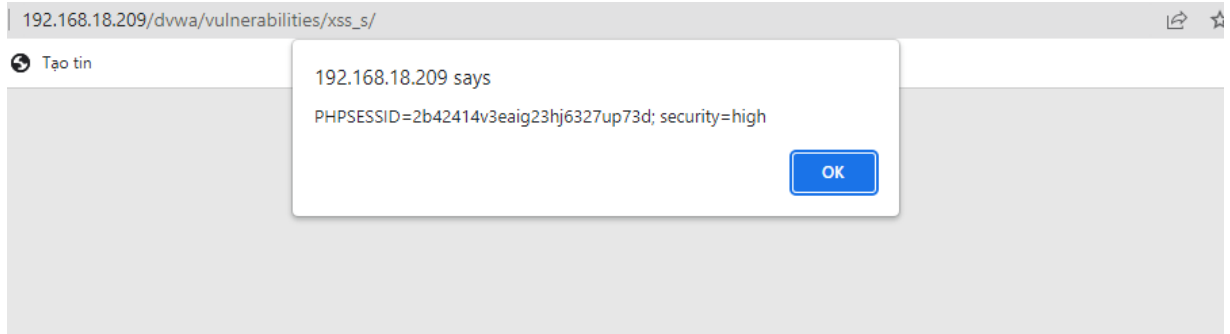
?>
```

Hình 52: Source code mức độ cao

Do ô name giới hạn bằng maxlength bằng 10 nên chúng em đã dùng html edit name thành maxlength bằng 50 để chèn được thẻ img. Do message không thể lấy được cookie hay bất kì thông tin nào mặc dù dùng thẻ nào đi chăng nữa. Nên em đã phát hiện ra ô name có lỗi hỏng nên chèn html vào để lấy cookie.



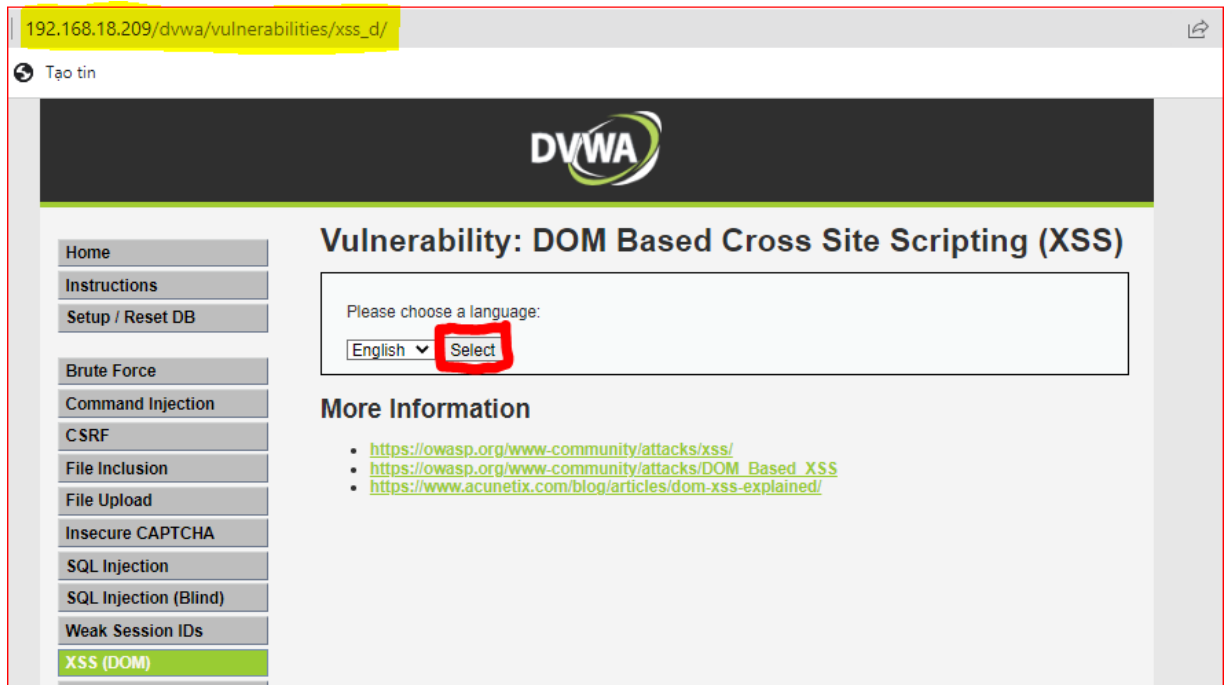
Hình 53: Chèn thẻ img vào trong ô name



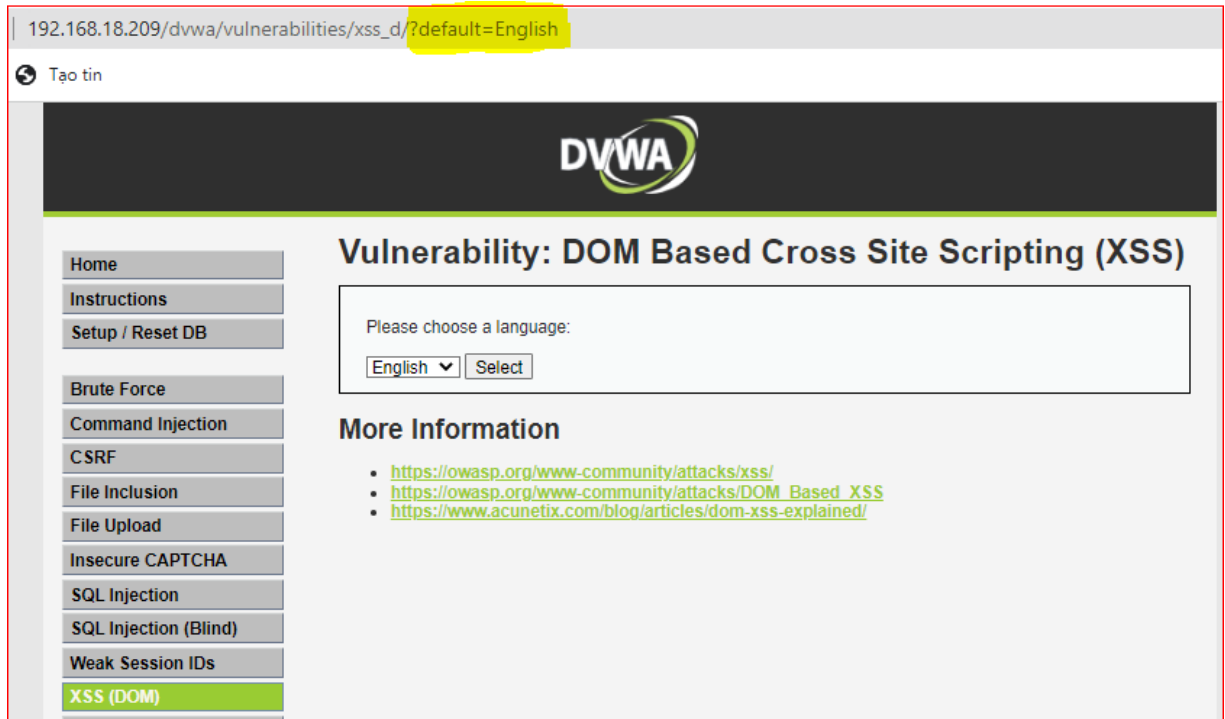
Hình 54: Lấy được cookie thông qua thẻ img

3.3. Demo Dom - based

Cách tấn công Dom – based XSS này mình sẽ tấn công thông qua thanh address bar. Ba sự lựa chọn demo sau đều có bước chung là thiết lập chế độ mặc định.



Hình 55: Trước khi click chuột chọn select

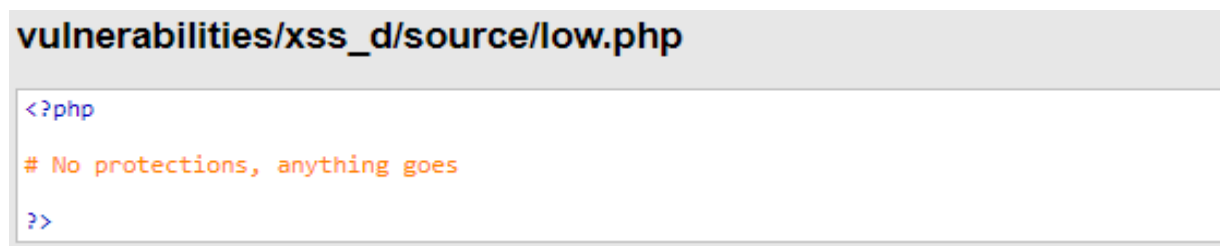


Hình 56: Sau khi chọn select thì address bar cũng thay đổi set chế độ default

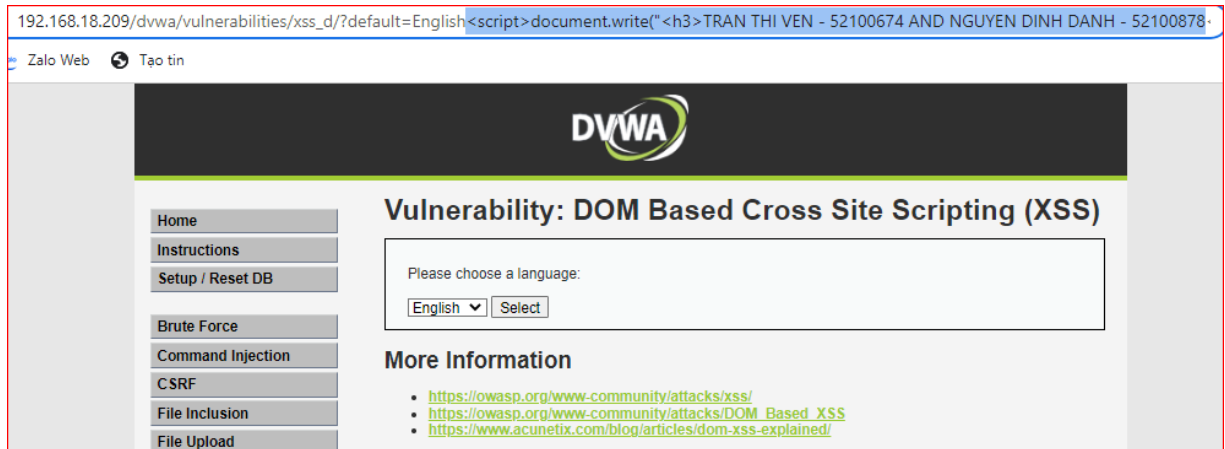
3.3.1. Mức độ an toàn thấp – Security Low

Sau khi set default cho ứng dụng thì tiếp tục chèn đoạn script vào sau địa chỉ này. Sau đó nhấn Enter.

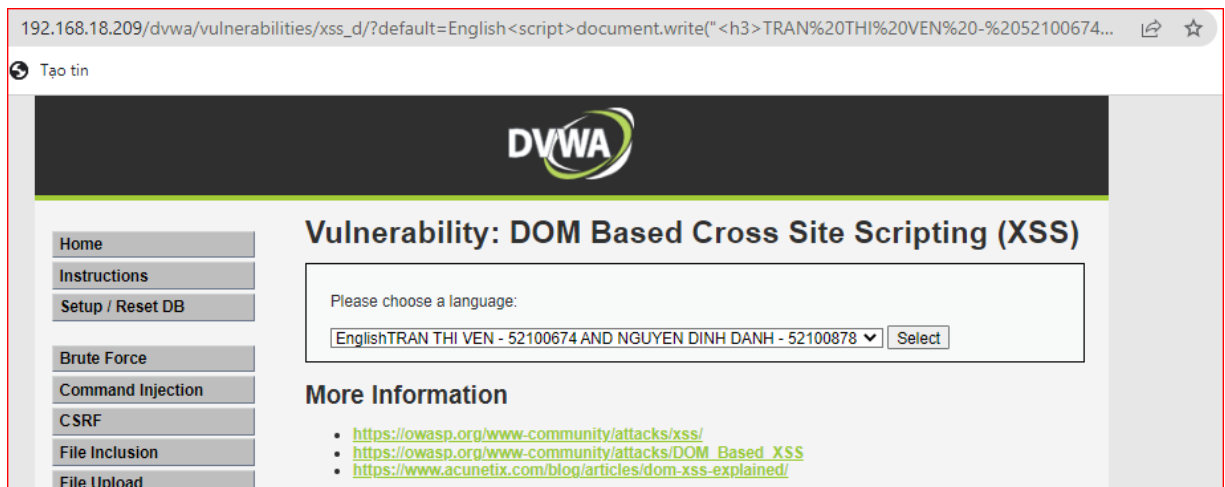
Đoạn script: `<script>document.write("<h3>TRAN THI VEN - 52100674 AND NGUYEN DINH DANH - 52100878</h3>");</script>`



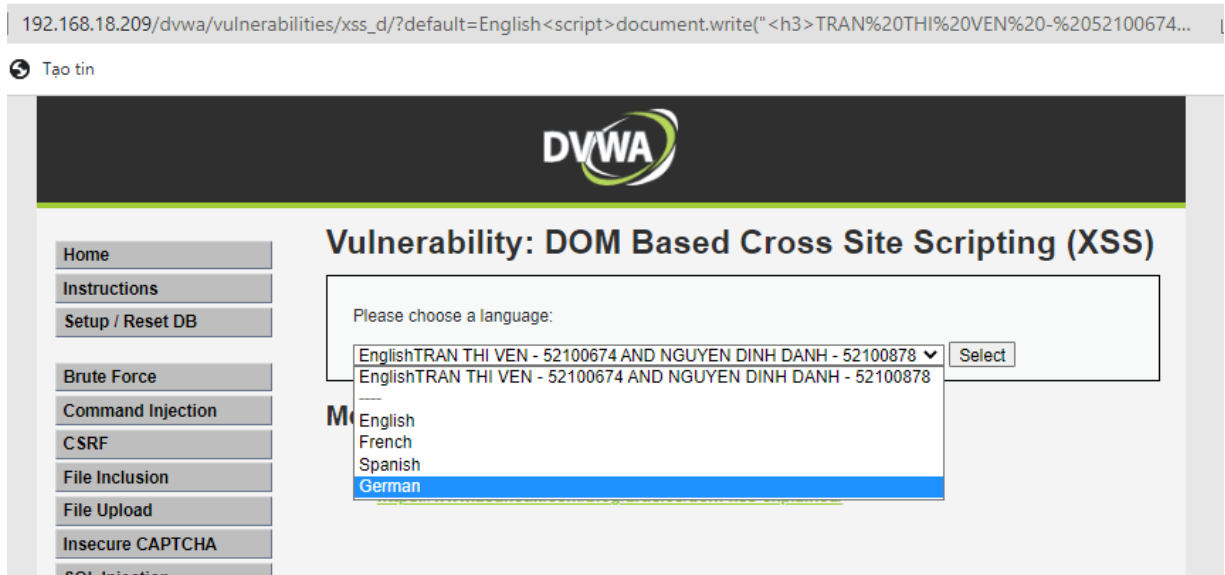
Hình 57: Source code của mức độ an toàn thấp



Hình 58: Chèn script vào address bar

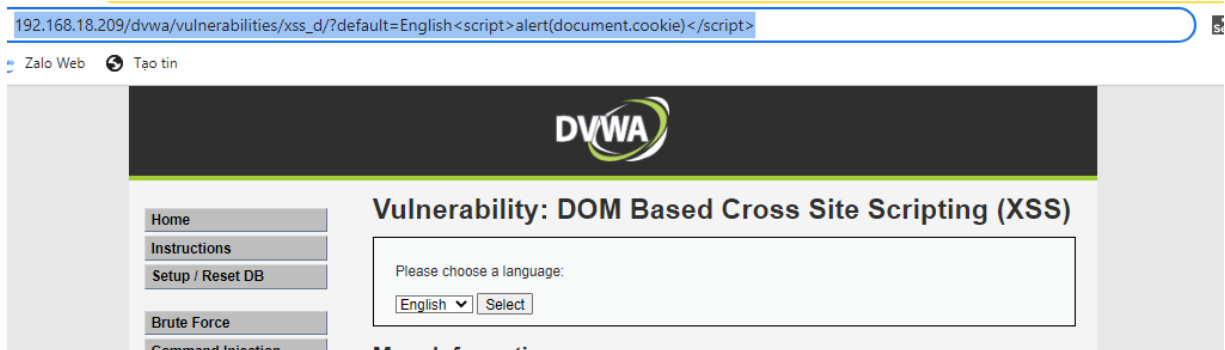


Hình 59: Sau khi đã nhấn enter và có được 1 ngôn ngữ mới xuất hiện

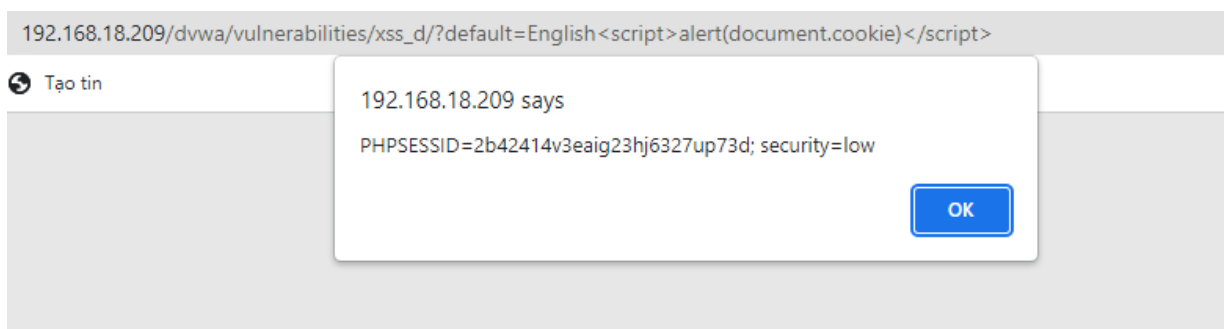


Hình 23: Được chọn thêm một ngôn ngữ mới

Ngoài ra chúng ta có thể get cookie của hệ thống ở mức độ thấp bằng cách chèn thẻ script như thẻ script ở trên: `<script>alert(document.cookie)</script>`



Hình 60: Source code ở mức độ thấp



Hình 61: Lấy được cookie

3.3.2. Mức độ an toàn trung bình và cao– Security Medium and High

vulnerabilities/xss_d/source/medium.php

```
<?php
// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {
    $default = $_GET['default'];

    # Do not allow script tags
    if (strpos ( $default, "<script" ) !== false) {
        header ( "location: ?default=English" );
        exit;
    }
}
?>
```

Hình 62: Source code mức độ trung bình

vulnerabilities/xss_d/source/high.php

```
<?php
// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {

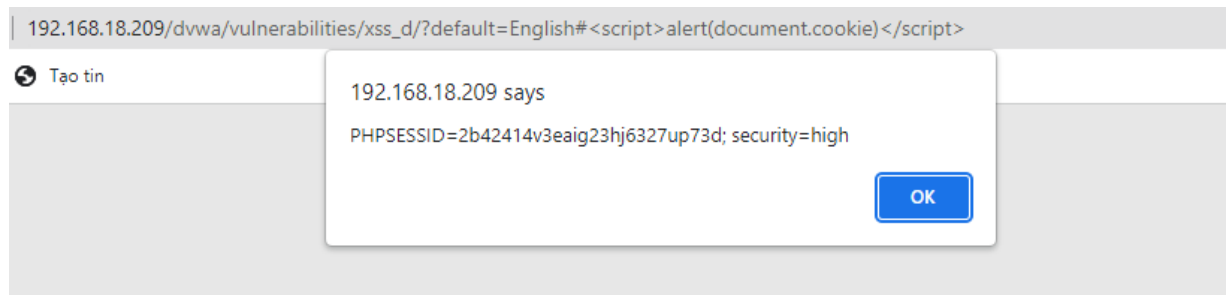
    # White list the allowable languages
    switch ( $_GET['default'] ) {
        case "French":
        case "English":
        case "German":
        case "Spanish":
            # ok
            break;
        default:
            header ( "location: ?default=English" );
            exit;
    }
}
?>
```

Hình 63: Source code mức độ cao

Vì mức độ code ở mức độ trung bình và cao đã mã hóa mã và các thẻ của mình chuyển thành false sau đó và get hẵn default. Nhưng chúng ta có thể mã hóa chúng bằng cách không cho lệnh của chúng ta gửi về máy chủ mà thực thi hẵn trên web server bằng câu lệnh: `#<script>alert(document.cookie)</script>`



Hình 64: Chèn dòng script lên trên thanh địa chỉ



Hình 65: Lấy được cookie ở mức độ cao