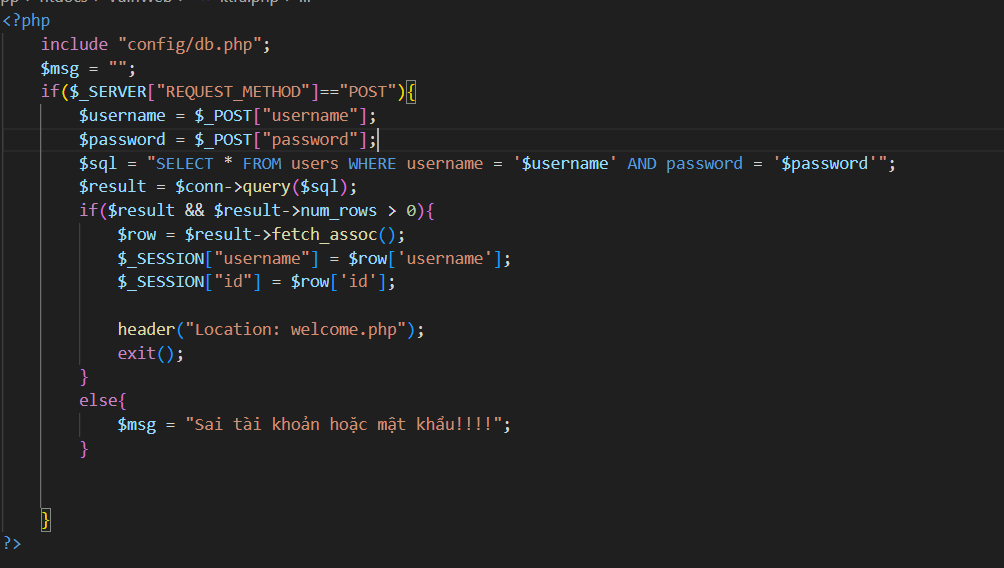
# **Báo cáo lỗ hổng web (DVWA)**

## **SQL Injection**

### *Mô tả*

SQL injection là lỗi chèn vào input người dùng câu lệnh truy vấn SQL mà ứng dụng không kiểm tra dữ liệu đầu vào.

### *Cách khai thác*

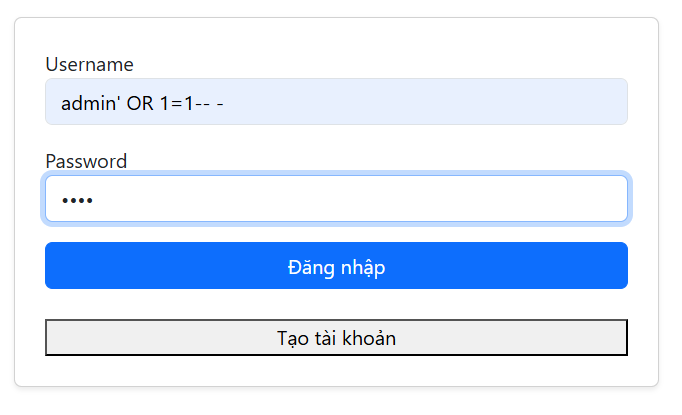


Đây là đoạn code không an toàn do input từ người dùng được đưa thẳng vào truy vấn SQL trên.

Do vậy, ta có thể khai thác chúng bằng câu lệnh truy vấn SQL để đăng nhập thông qua câu lệnh truy vấn “ ‘ OR 1=1-- ” và nhập passwd tùy ý và câu lệnh sql sẽ trở thành SELECT \* FROM users WHERE username = ‘ ‘ OR 1=1-- password = ‘abc’.

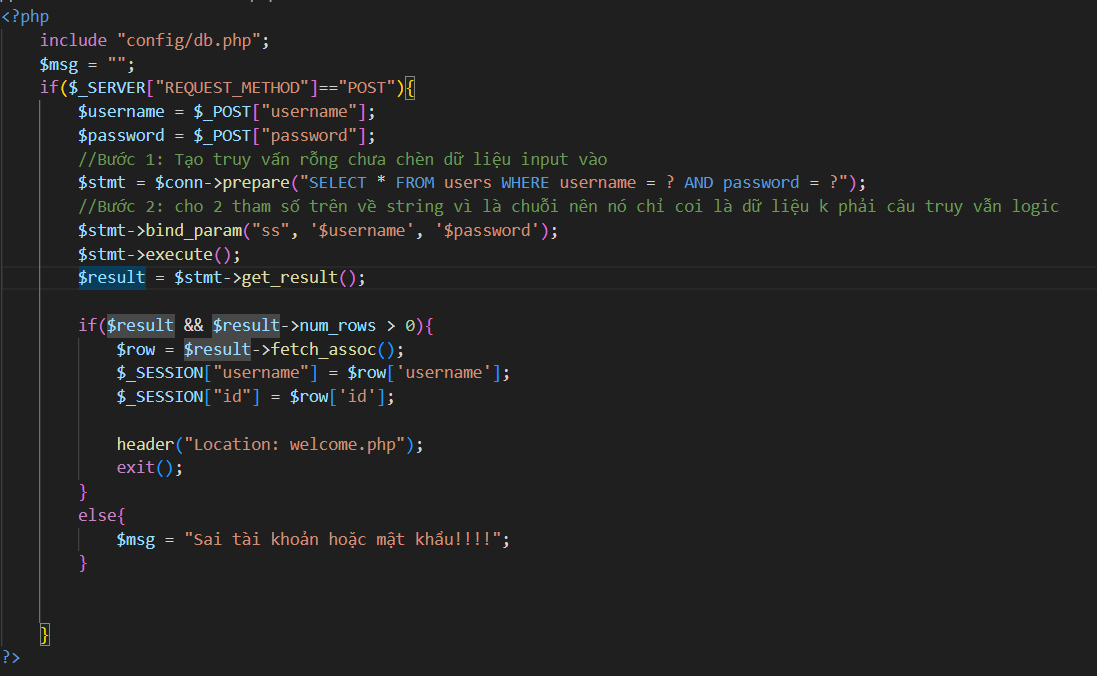
Trường hợp OR 1=1 là luôn đúng và “--” là dấu cmt nên nó sẽ bỏ qua kiểm tra mật khâu -> login.

### *Kết quả*





### *Cách fix lỗi*



Sử dụng prepared statement là phương thức để thực thi câu truy vấn SQL một cách an toàn, sử dụng placeholder xong liên kết các giá trị vào chúng trước khi thực thi.

Tạo câu truy vấn với giá trị rỗng sau đó dùng bind\_param để gán giá trị username và password vào truy vấn trên nhưng bind\_param biến 2 giá trị trên là string nên khi cho vào câu truy vấn thì nó chỉ là dữ liệu chứ không phải câu lệnh truy vấn.

## **IDOR (Insecured Direct Object References)**

### *Mô tả*

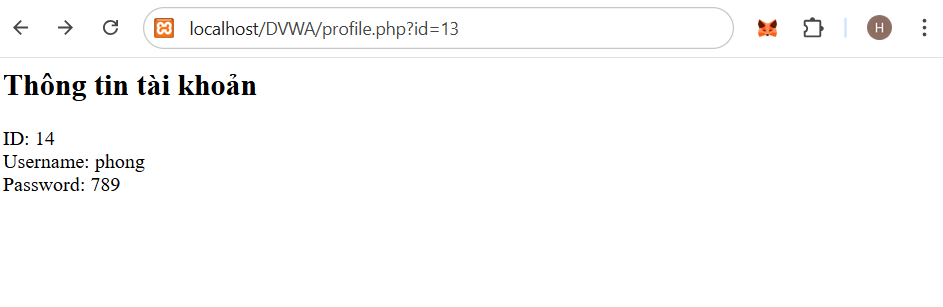
IDOR là lỗi xảy ra khi chương trình cho phép người dùng truy cập tài nguyên 1 cách bất hợp pháp, thông qua dữ liệu do người dùng cung cấp.

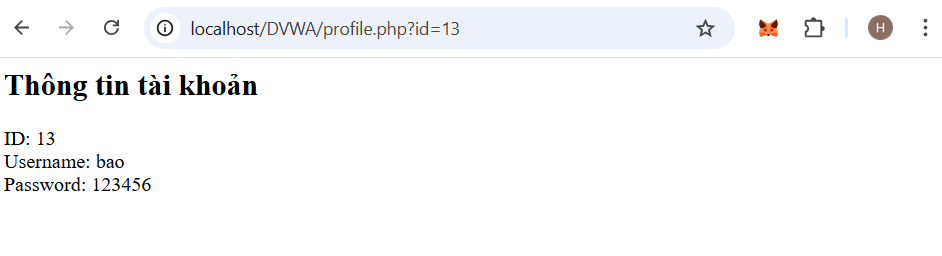
### *Cách khai thác*

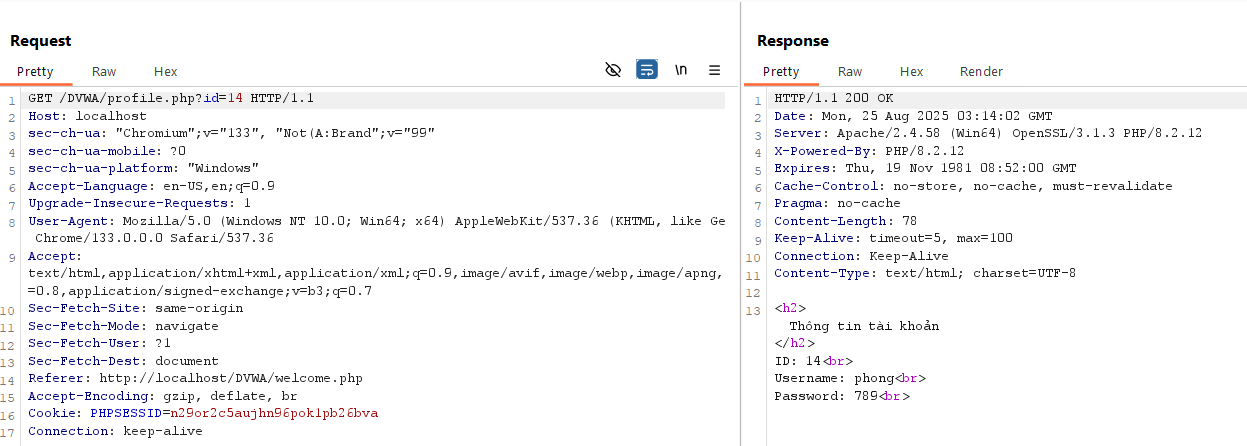


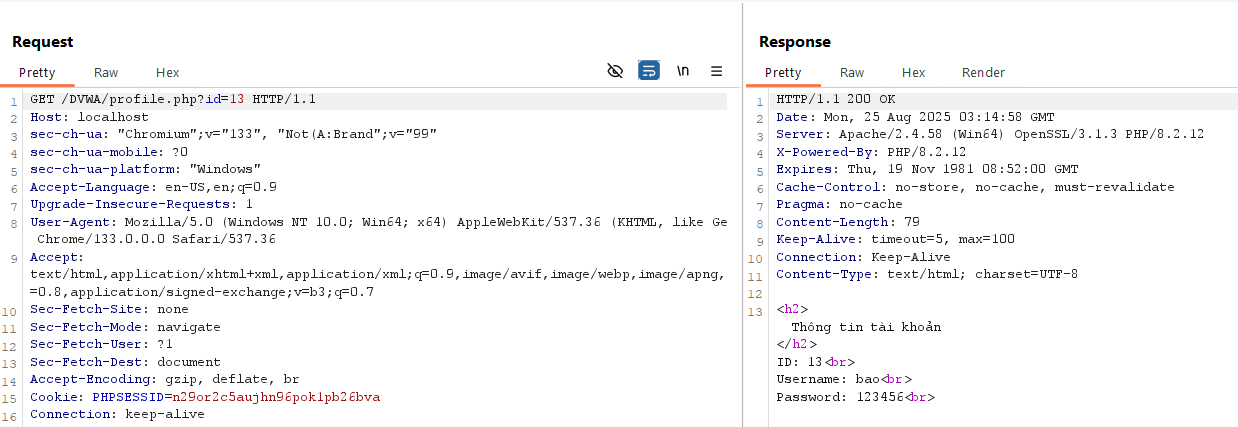
Đây là đoạn code gây lỗi do sử dụng phương thức get mà không có biện pháp kiểm soát nên bất cứ ai cũng có thể thay đổi ID và xem những thông tin không được quyền xem của người khác. Chỉ cần chỉnh sửa url thay đổi ID như sau: “localhost/DVWA/profile.php?id=…” là có thể truy cập vào ID người dùng khác.

### *Kết quả*









Dùng phương thức get không kiểm tra session và id nên khi thay đổi id thì server vẫn ngầm hiểu rằng là mình đã đăng nhập thành công rồi nên trả về dữ liệu của get id đó. Như ảnh trên cho thấy cookie của chúng giống nhau nên vẫn server hiểu rằng người dùng đăng nhập thành công nên khi thay đổi id thì server vẫn trả về dữ liệu.

### *Cách fix lỗi*



Do đó, ta sẽ sửa GET[‘id’] thành SESSION[‘id’] để khi đăng nhập thì session sẽ lưu thông tin id cố định theo người đăng nhập thành công nên khi có chỉnh sửa url thì không cho kết quả vì nó bỏ qua không dùng GET[‘id’].





SESSION[‘id’] = 14 nên khi đổi id =13 thì nó k hiển thị của user có id là 13

## **Stored XSS**

### *Mô tả*

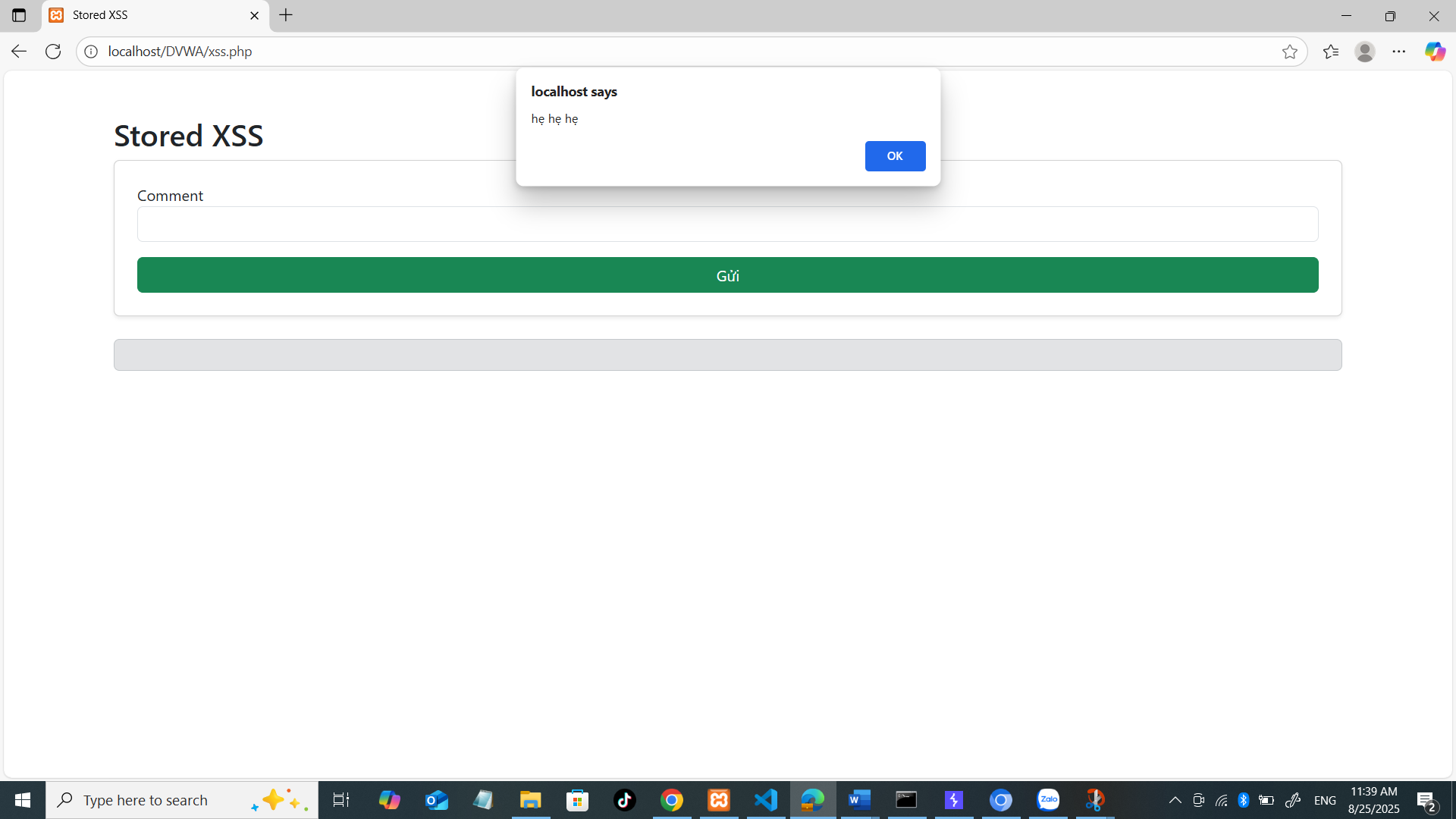
Stored XSS là 1 loại của tấn công XSS nơi mà kẻ tấn công đầu tiên gửi đoạn mã tới ứng dụng web, sau đó lưu trữ đoạn mã như vào cơ sơ sở dữ liệu hay tệp nội dung phía server, và cuối cùng ứng dụng thực thi nó một cách vô ý cho mỗi nạn nhân ghé vào trang web của nó.

### *Cách khai thác*

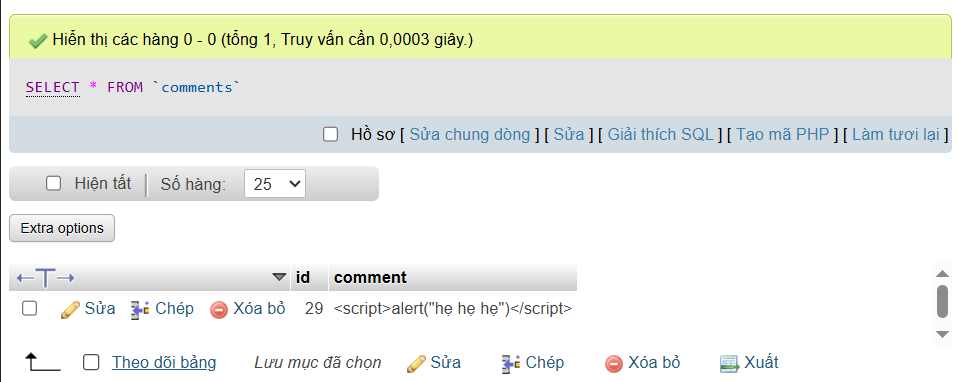


Đây là đoạn code lỗi do không lọc input nên khi gửi comment thì nó sẽ thực hiện câu truy vấn để chèn comment vào trong cơ sở dữ liệu là bẳng comments. Sau đó, khi hiển thị in đoạn comment thì chương trình vô tình thực hiện đoạn mã độc đó chứ không phải hiển thị text. Payload có thể chèn vào như sau: “<script>alert("hẹ hẹ hẹ")</script>.

### *Kết quả*



Đây là kết quả khi chèn đoạn payload vào dẫn đến trang web hiển thị đoạn payload.

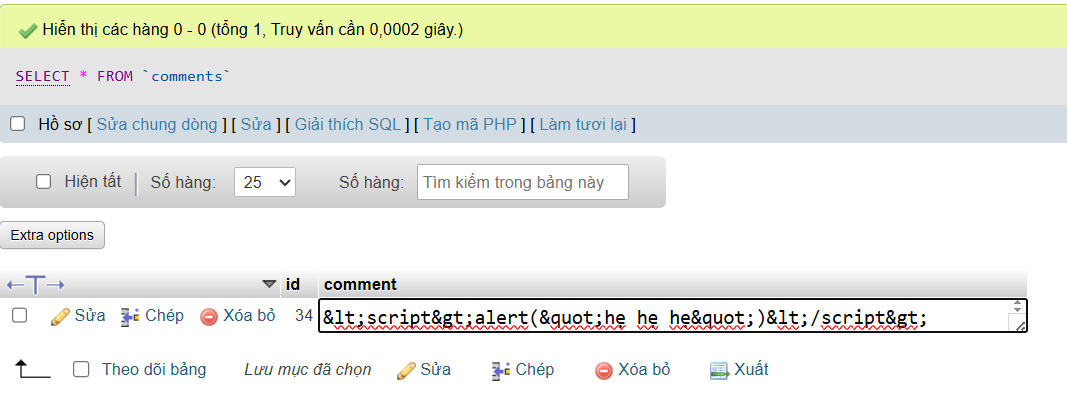


Do đoạn payload được lưu vào trong database nên khi bất cứ ai ghé vào trang web thì nó sẽ tự dộng thực thi đoạn mã đó.

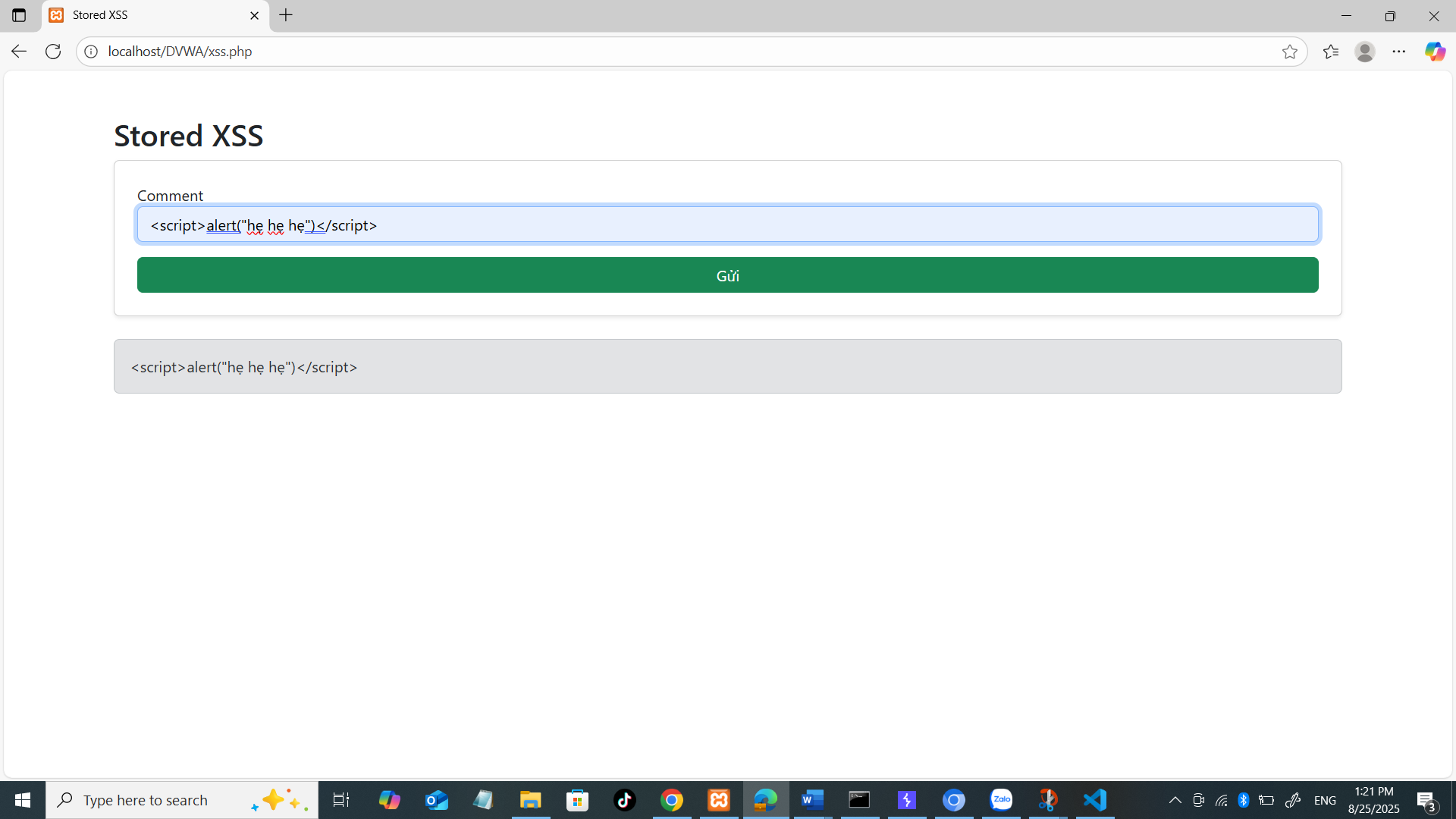
### *Cách fix lỗi*



Ta lọc input sử dụng htmlspecialchars để biến đoạn payload độc hại thành đoạn text bình thường nó sẽ chuyển “& → &amp, < → &lt, > → &gt” , encode ent\_quotes dấu ‘’.



Khi lưu vào trong cơ sở dữ liệu thì nó đã được encode về ký tự thường nên không thể thực thi được đoạn payload đó.



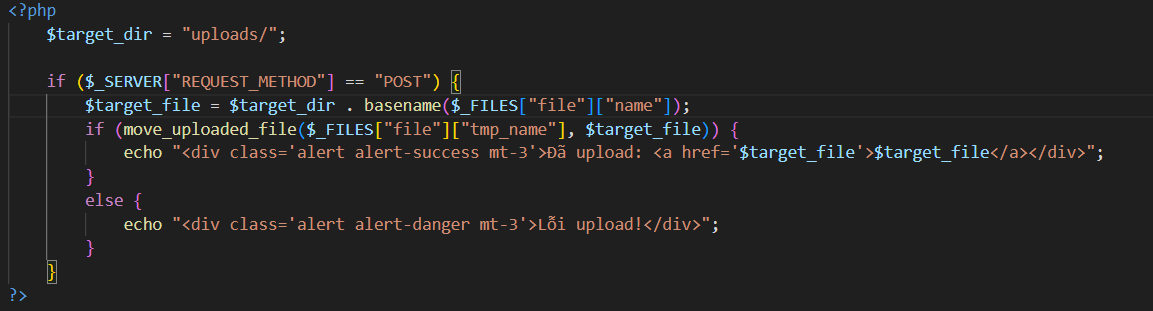
Nhưng khi hiển thị trên trang web thì nó đc decode thành dạng hiển thị cho người dùng dễ đọc.

## **File Upload**

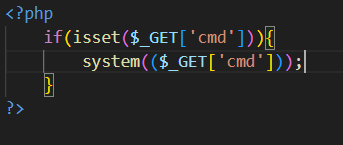
### *Mô tả*

Lỗi file upload là khi một web server cho phép người dùng tải file lên file hệ thống của nó mà không kiểm tra lọc tên file, loại file, nội dung hay kích cỡ.

### *Cách khai thác*

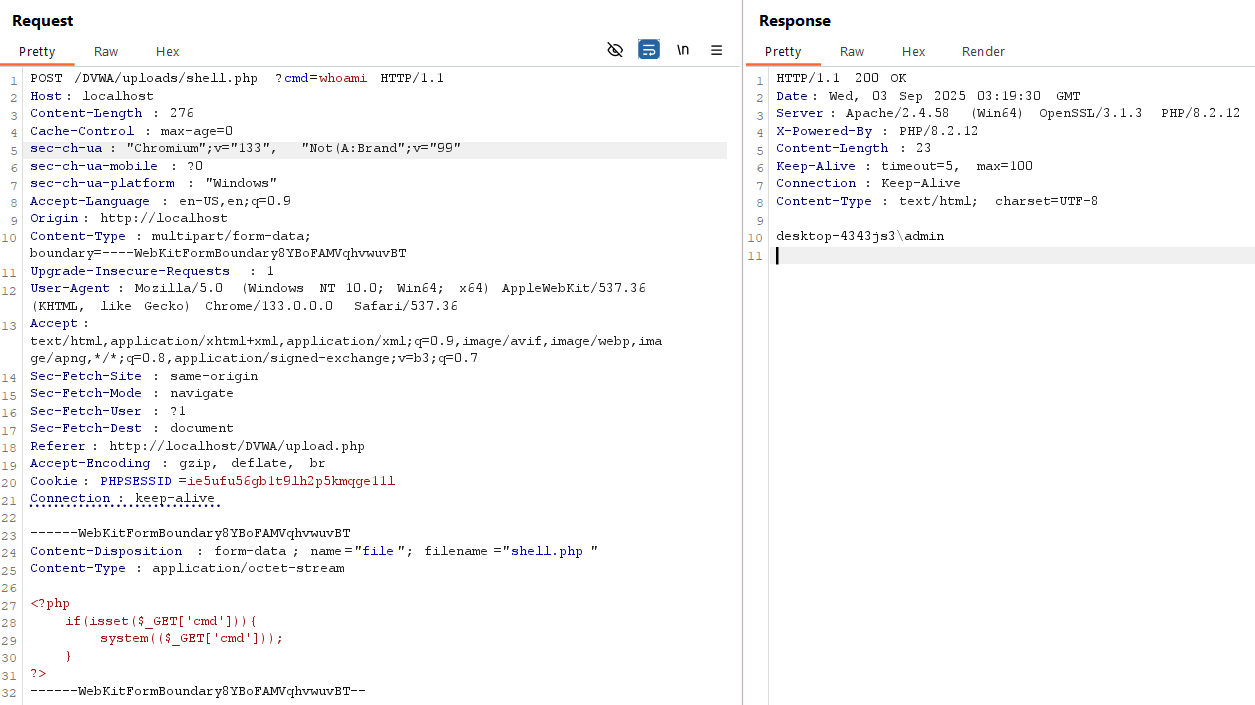


Đây là đoạn code lỗi do không thực thực hiện kiểm tra file mà lưu thẳng file vào thư mục uploads dẫn đến kẻ tấn công có thể upload 1 file độc hại lên chương trình như sau:

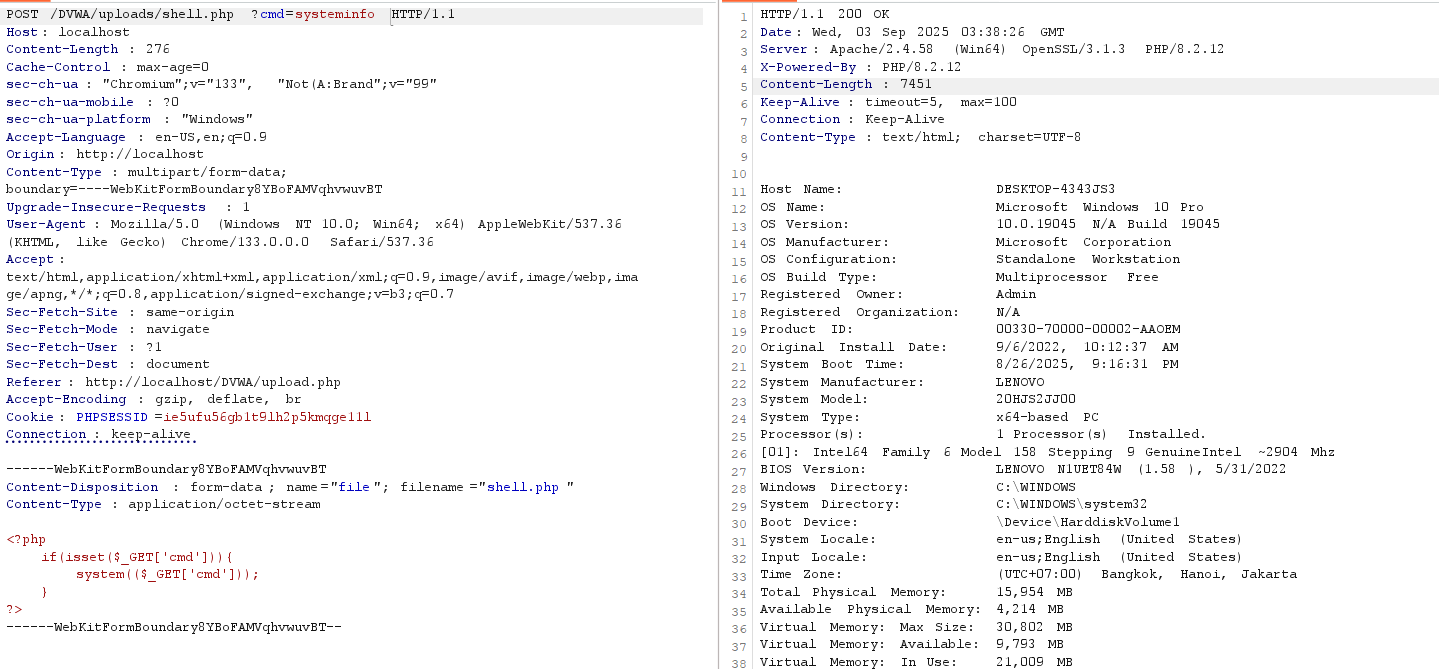


Đoạn nội dung file độc hại trên là kiểm tra có xem url có cmd hay không, hệ thống thực thi câu lệnh mà truyền qua cmd => kết quả sẽ hiển thị ở trong trình duyệt.

### *Kết quả*



Thử xem quyền người dùng kết quả trên cho thấy máy đang là người dùng admin.



Xem thông tin hệ thống.

### *Cách fix lỗi*



Tạo whitelist chỉ cho phép tải những tệp tin được thiết lập sẵn lên.

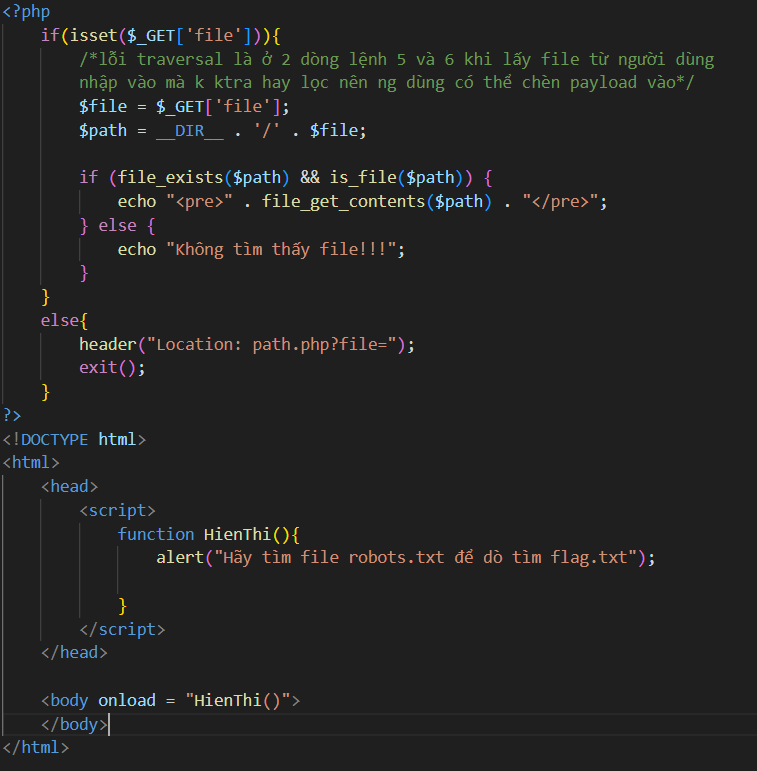


## **Path Traversal**

### *Mô tả*

Path traversal là 1 ứng dụng web lỗ hổng mà cho phép kẻ tấn công truy cập các tệp tin hoặc thư mục như là các tệp hệ thống giới hạn hay mã nguồn, bằng cách khai thác xác thực đầu vào không đầy đủ.

### *Cách khai thác*

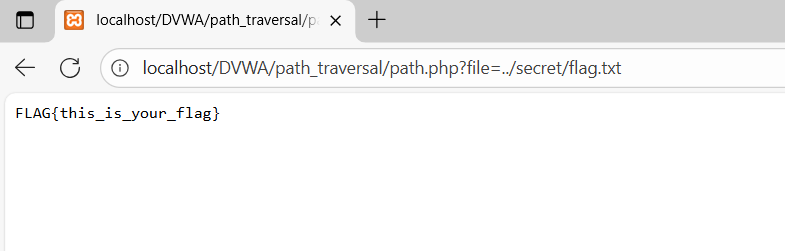


Đoạn code trên lấy file ghép vào đường dẫn mà không kiểm tra đầu vào nên có thể chèn payload vào

VD: file: ../secret/flag.txt, path: <http://localhost/DVWA/path_traversal/path.php?file=>

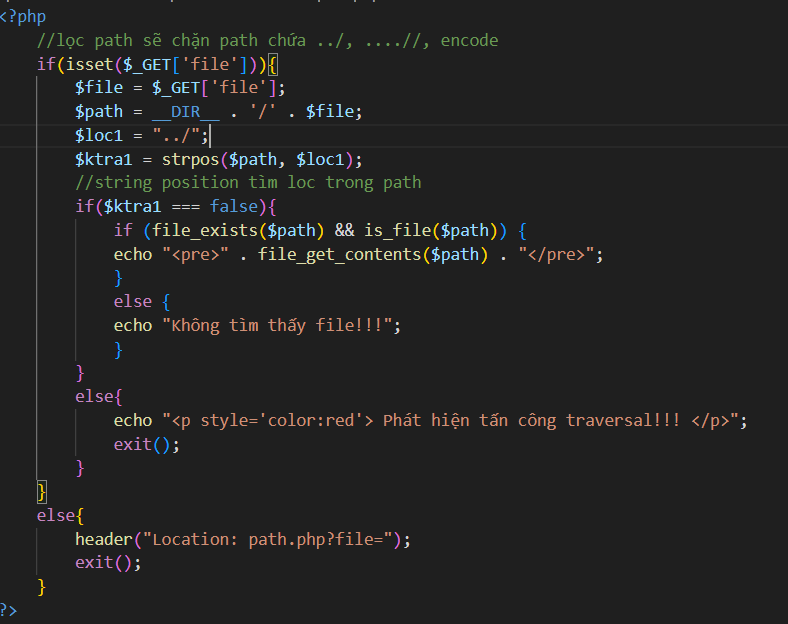
Do không kiểm tra nên nó sẽ ghép vào thành <http://localhost/DVWA/path_traversal/path.php?file=../secret/flag.txt> và thực thi hiển thị file flag đó

### *Kết quả*

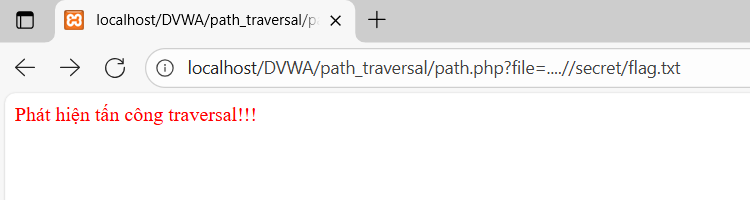


File flag.txt nó được hiển thị điều dẫn đến có thể bị lộ dữ liệu

### *Cách fix lỗi*



Mình sẽ lọc đầu vào sẽ loại bỏ các đường dẫn không hợp lệ như “../”, “….//”, encode url. Do đó, khi chỉ cần thấy trong chuỗi đường dẫn có các ký tự lọc đó thì sẽ không thực thi hiển thị nội dung file.

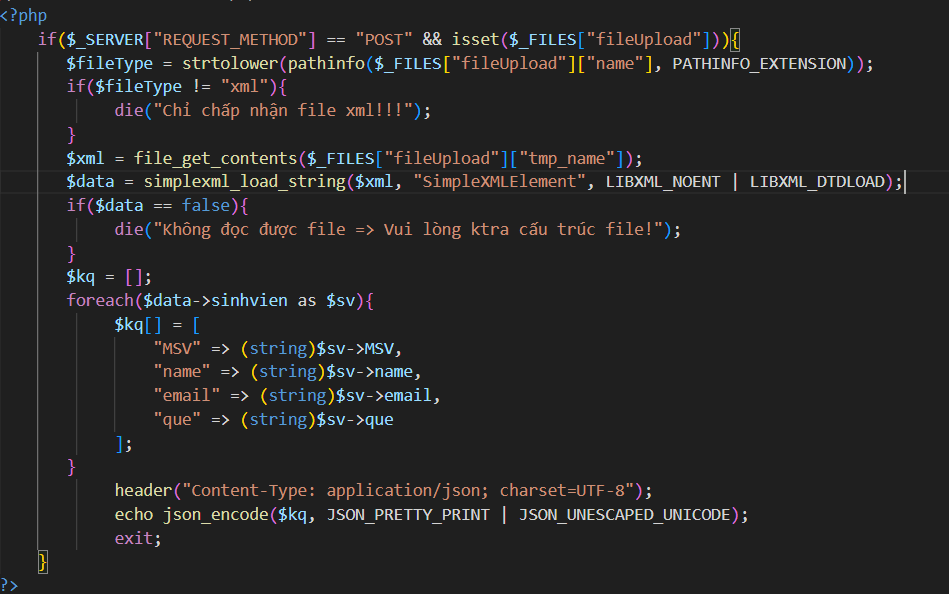


## **XXE Injection (XML External Entity)**

### *Mô tả*

XXE xảy ra khi ứng dụng xử lý dữ liệu XML không cấu hình đúng parser, cho phép kẻ tấn công chèn và khai thác external entity trong XML.

### *Cách khai thác*



Đoạn code lỗi XXE là đoạn biến dữ liệu $data có 2 biến cờ cuối là thay thế entity trong xml thành nội dung thực của nó và cho phép tải xử lý DTD (Document Type Definition) dẫn đến có thể chèn vào XML 1 đoạn DTD định nghĩa entity bên ngoài để trỏ tới file hệ thống.

VD: <!DOCTYPE foo[

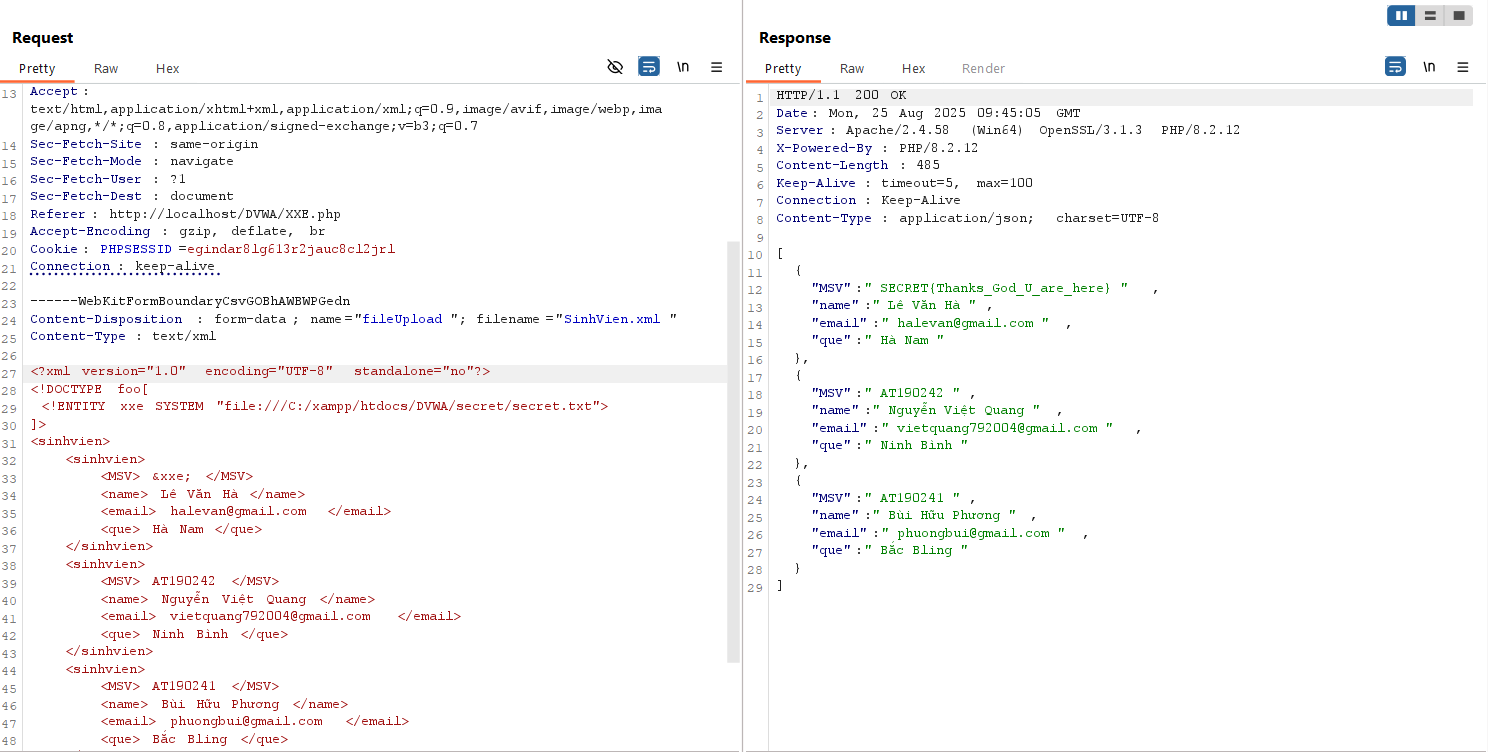
<!ENTITY xxe SYSTEM “file:///C:/xampp/htdocs/DVWA/secret/secret.txt”>

]>

<>&xxe;<>

Khi xử lý 1 document và định nghĩa external entity xxe thì nó sẽ nội dung trong xml thành nội dung thực của external entity và khi gọi nó sẽ thực thi hiển thị file trong phần xml.

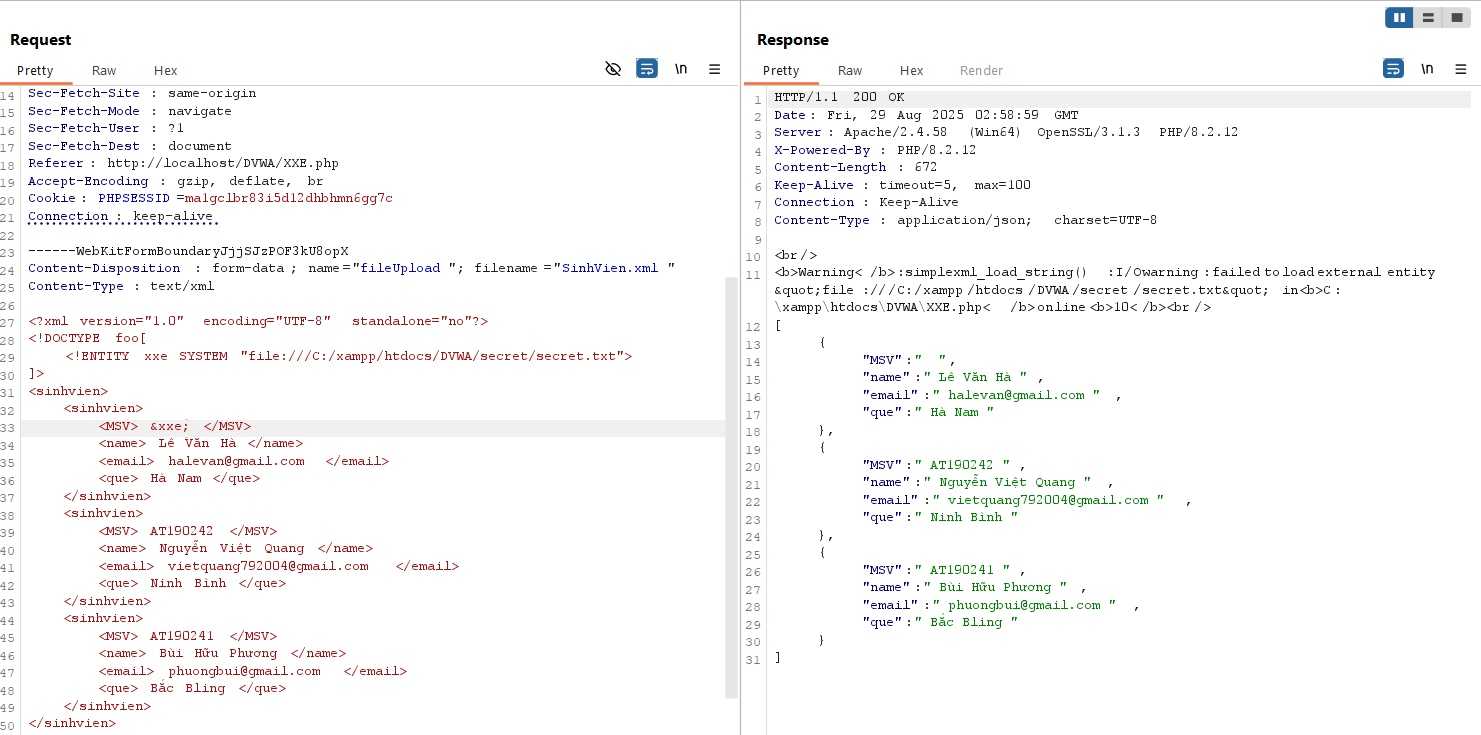
### *Kết quả*



### *Cách fix lỗi*



Chặn không cho load entity thì nó sẽ không thể thực thi nội dung của entity và hiển thi ra xml được.



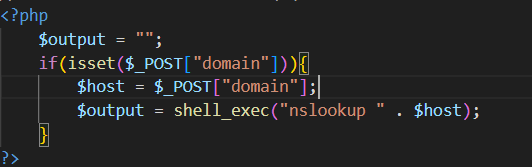
## **Command Injection**

### *Mô tả*

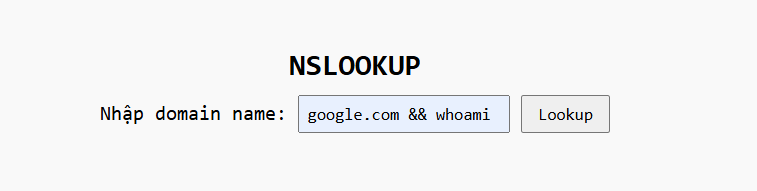
Command Injection cho phép kẻ tấn công thực thi câu lệnh hệ thống tùy ý trên thiết bị dẫn đến hoàn toàn xâm nhập vào hệ thống mục tiêu. Nó xảy ra khi chương trình không kiểm tra đầu vào người dùng. Lỗi này cho phép thực hiện những câu lệnh độc hại.

### *Cách khai thác*

Lỗi này khai thác bằng các câu lệnh cmd như whoami, cat để đọc file, ls để hiển thị các file có trong máy, …



Đây là đoạn code lỗi do khi người dùng nhập input vào chương trình không kiểm tra input nên khi post domain lên thì thực thi câu lệnh truy vấn tên miền luôn dẫn đến kẻ tấn công có thể tiêm những câu lệnh độc hại vào chương trình.



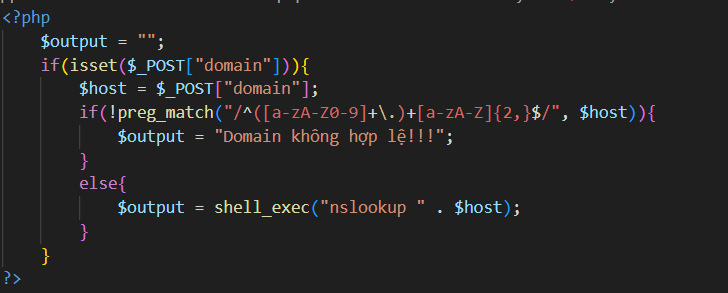
Trong trường hợp này chương trình truy vấn thông tin têm miền DNS khi thực hiện truy vấn thông thường như google.com để xem thông tin về tên miền này nhưng ta kèm theo câu lệnh whoami để xem mình đang là quyền gì.

### *Kết quả*

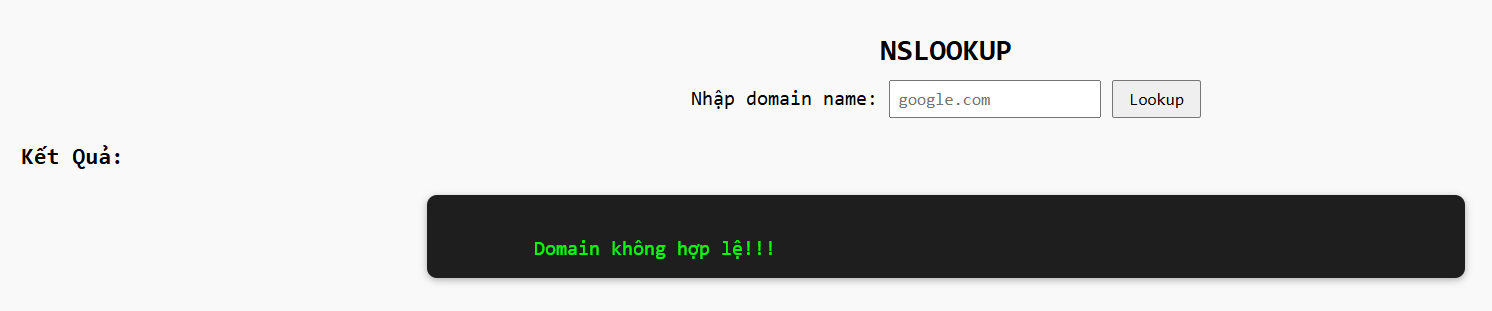


Kẻ tấn công thực hiện truy vấn trên cho kết quả thông tin DNS cùng với tên máy và quyền người dùng. Thông tin này cho phép kẻ tấn công có đường hướng tấn công tiếp theo sau đó.

### *Cách fix lỗi*



Thêm câu lệnh ktra đầu vào host domain theo dạng form quy định như trên là tên miền cho phép là chuỗi ký tự thường, hoa và số sau đó là đấu chấm, đuôi phải ít nhất từ 2 ký tự trở lên là các ký tự thường hoặc hoa như com, net, vn,…

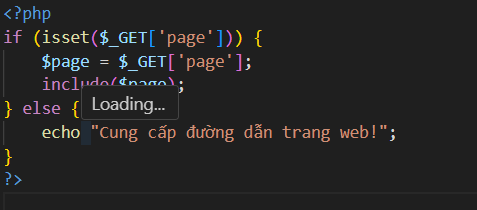


## **Remote File Inclusion**

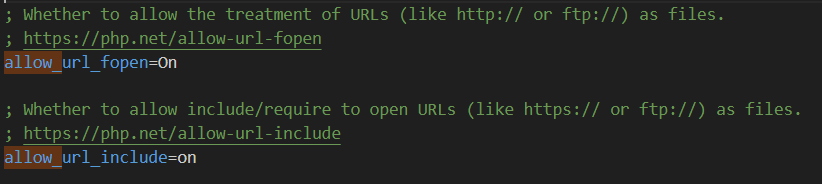
### *Mô tả*

RFI là lỗi cho phép kẻ tấn công nhúng 1 mã độc hại đc tùy chỉnh trên trang web hoặc máy chủ bằng cách sử dụng các tập lệnh. Nó còn cho phép tải lên 1 tệp nẳm trên máy chủ khác chuyển đến dưới dạng hàm PHP (include, include\_once, require, require\_once).

### *Cách khai thác*

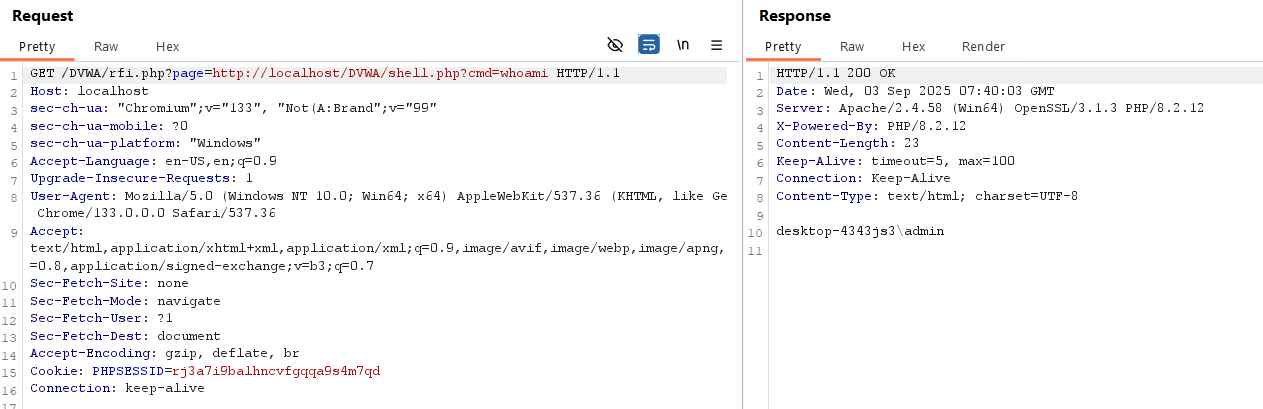


Set giá trị hàm allow\_url\_include = on



Để thực hiện cuộc tấn công này, chúng ta thử nhúng các đường dẫn vào trang web có sẵn: <http://localhost/DVWA/rfi.php?page=http://localhost/DVWA/shell.php?cmd=whoami> gọi đến đường dẫn khác có thể là của 1 server khác tùy ý link đến đoạn code độc hại để thực thi nó.

### *Kết quả*



### *Cách fix lỗi*

Có thể dùng whitelist cho các file extension được cho phép.

Set allow\_url\_fopen và allow\_url\_include thành off để giới hạn việc có thể gọi các tệp tin từ xa.