bWAPP

HTML Injection – Stored (Blog)

Tổng quan về HTML Injection:

- HTML là gì?

HTML là một ngôn ngữ đánh dấu mà tất cả các element của website đều được viết trong các thẻ. Nó hầu hết được sử dụng để tạo website. Các trang web sẽ được gửi tới trình duyệt dưới dạng các tài liệu HTML. Sau đó, các tài liệu HTML đó được convert sang website thông thường và hiển thị cho người dùng.

- HTML Injection là gì?

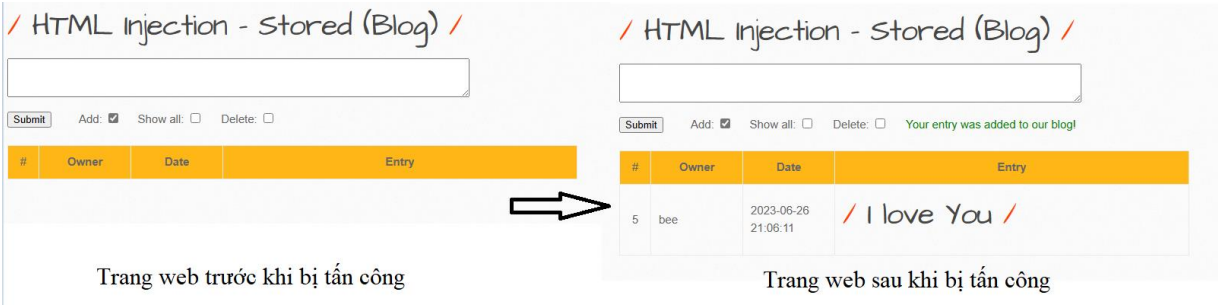
HTML Injection là một loại tấn công mà hacker sẽ tiêm code HTML vào website thông qua những lỗ hổng, với mục đích thay đổi thiết kế hoặc một số thông tin của website, rồi hiển thị cho user. Kết quả user có thể nhìn thấy những dữ liệu mà hacker đã gửi vào

Những dữ liệu này sẽ khác nhau dựa vào loại tấn công. Nó có thể là 1 vài thẻ HTML, cũng có thể là 1 form hoặc 1 trang web fake. Khi tấn công xảy ra, trình duyệt sẽ hiển thị dữ liệu mà hacker đã tiêm vào.

Thay đổi hiển thị website không phải là rủi ro duy nhất khi cuộc tấn công này xảy ra. Nó cũng tương tự cuộc tấn công XSS, nên hacker có thể dễ dàng lấy cắp danh tính của người dùng.

- HTML Injection – Stored (Blog) xảy ra khi nào

Stored HTML injection xảy ra khi đoạn mã HTML độc hại được lưu trữ vào web server và được thực thi mỗi khi user gọi một tính năng phù hợp. Loại tấn công này thường ảnh hưởng đến nhiều người dùng.



Trang web trước khi bị tấn công

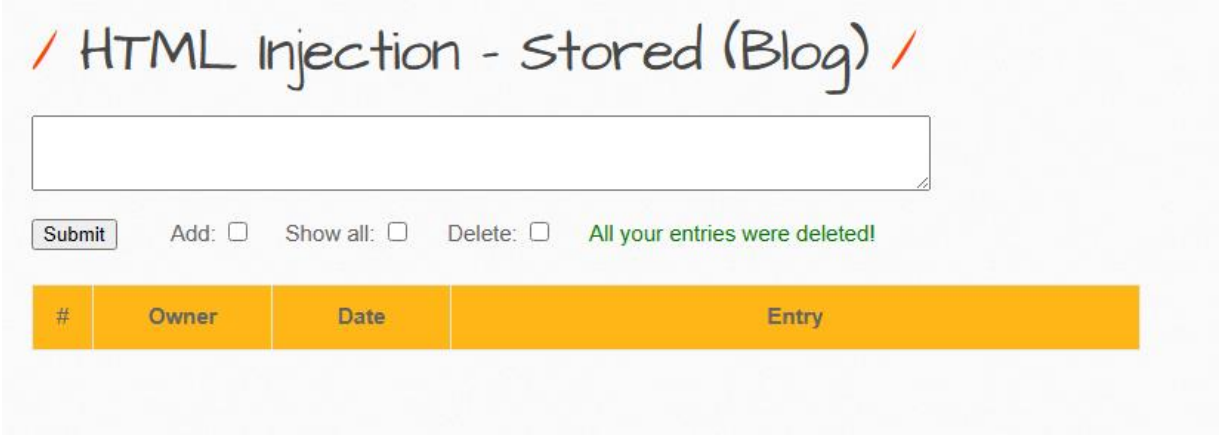
Trang web sau khi bị tấn công

Hình ảnh minh họa

Và sau đây chúng ta sẽ tấn công HTML injection – Stored (Blog) với bWAPP

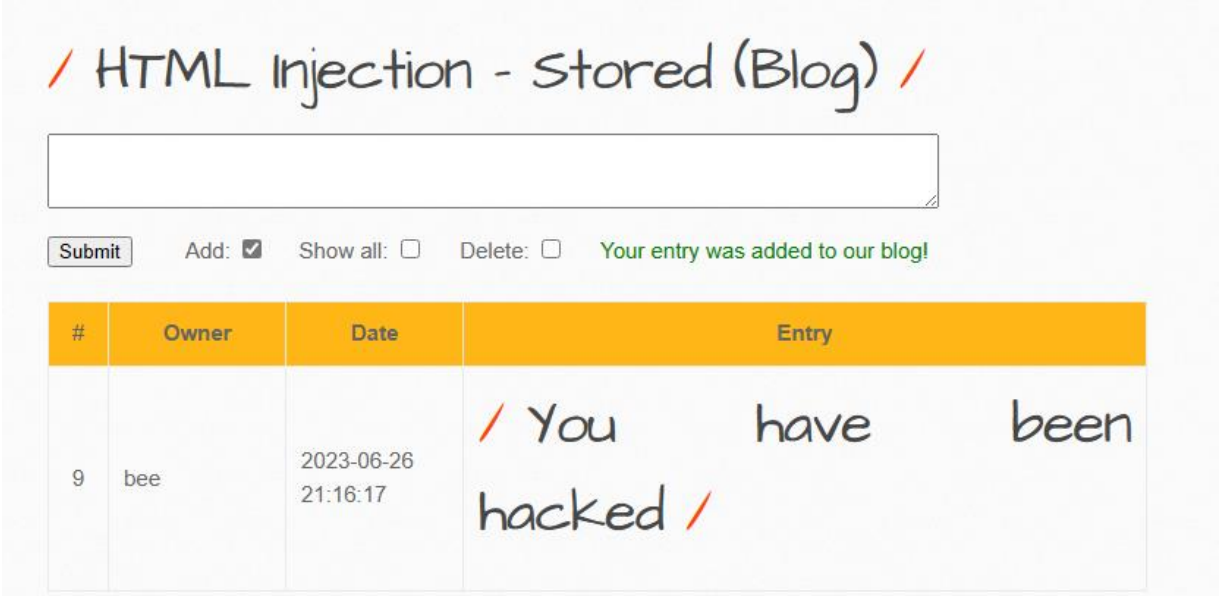
- Level Low

Đây là hình ảnh ban đầu của trang web mà ta muốn tấn công



Hình ảnh ban đầu của trang web

Bây giờ ta sẽ thử tiêm một câu lệnh HTML ví dụ như <h1>You have been hacked</h1>



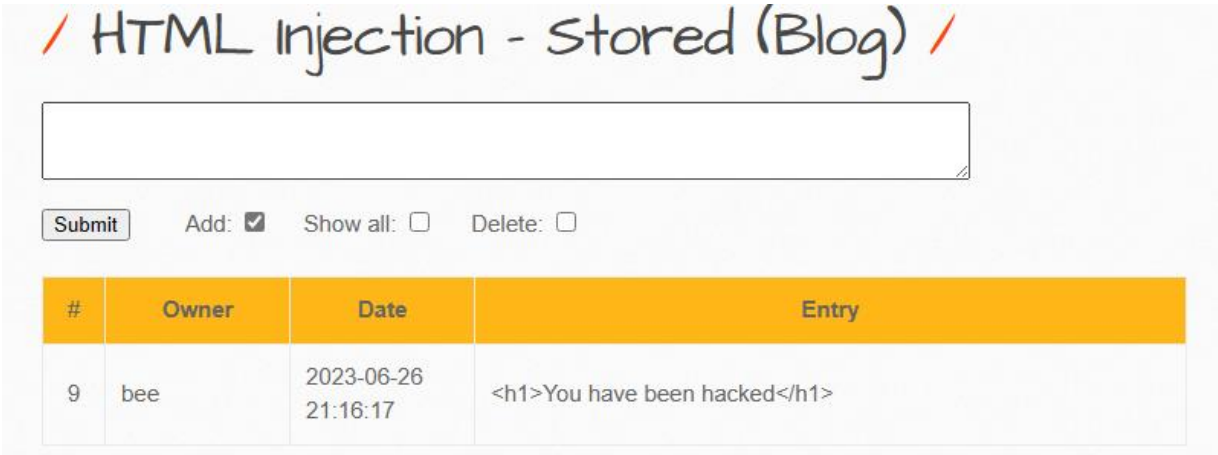
Hình ảnh sau khi tấn công

Và đây là kết quả chúng ta nhận lại được, thể hiện chúng ta đã thành công tiêm mã HTML vào trang web với level Low

- Level Medium

Tiếp theo ta sẽ đi đến với level Medium của bWAPP

Chúng ta đã thấy được trang web đã bắt đầu lọc đầu vào và những kí tự của chúng ta đã nhập vào đã hiển thị hết lên trên web



Hình ảnh sau khi chuyển sang level Medium

Chúng ta sẽ chuyển đến phần code của trang web này

```
switch($_COOKIE["security_level"]) {
        case "0" :
            $data = sqli_check_3($link, $data);
            break;

        case "1" :
            $data = sqli_check_3($link, $data);
            // $data = xss_check_4($data);
            break;

        case "2" :
            $data = sqli_check_3($link, $data);
            // $data = xss_check_3($data);
            break;
        default :
            $data = sqli_check_3($link, $data);
            break;
    }
```

Đây là phần code chọn level của trang web cho thấy được khi chúng ta chọn level là medium sẽ sử dụng hàm check là xss_check_4 và level high là hàm xss_check_3. Nhưng mà ở bên trong file htmli_stored.php thì lại có một đoạn code như sau

```
 if($_COOKIE["security_level"] == "1" or $_COOKIE["security_level"] == "2")
   {
?>
      <tr height="40">
         <td align="center"><?php echo $row->id; ?></td>
         <td><?php echo $row->owner; ?></td>
         <td><?php echo $row->date; ?></td>
         <td><?php echo xss_check_3($row->entry); ?></td>
      </tr>
```

Ở những dòng code này đang thể hiện nếu level chúng ta chọn là medium hay high thì đều sử dụng hàm check là xss_check_3 nên chúng ta sẽ đổi một chút ở phần code này để cho đúng với đề bài

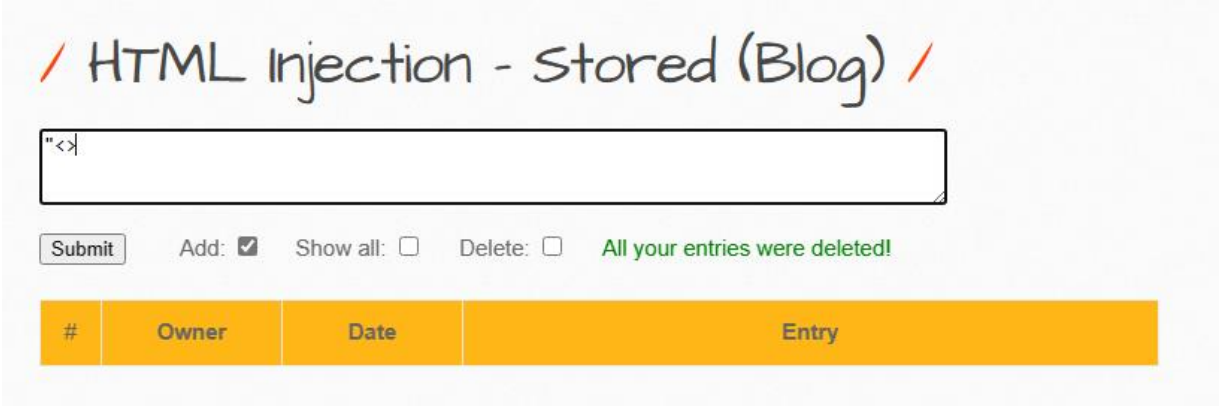
```
if($_COOKIE["security_level"] == "1" or $_COOKIE["security_level"] == "2")
   {
?>
      <tr height="40">
         <td align="center"><?php echo $row->id; ?></td>
         <td><?php echo $row->owner; ?></td>
         <td><?php echo $row->date; ?></td>
         <td><?php echo xss_check_4($row->entry); ?></td>
      </tr>
```

Như vậy là giờ ta sẽ thử với hàm check xss_check_4 giờ thi tiếp tục công việc của chúng ta

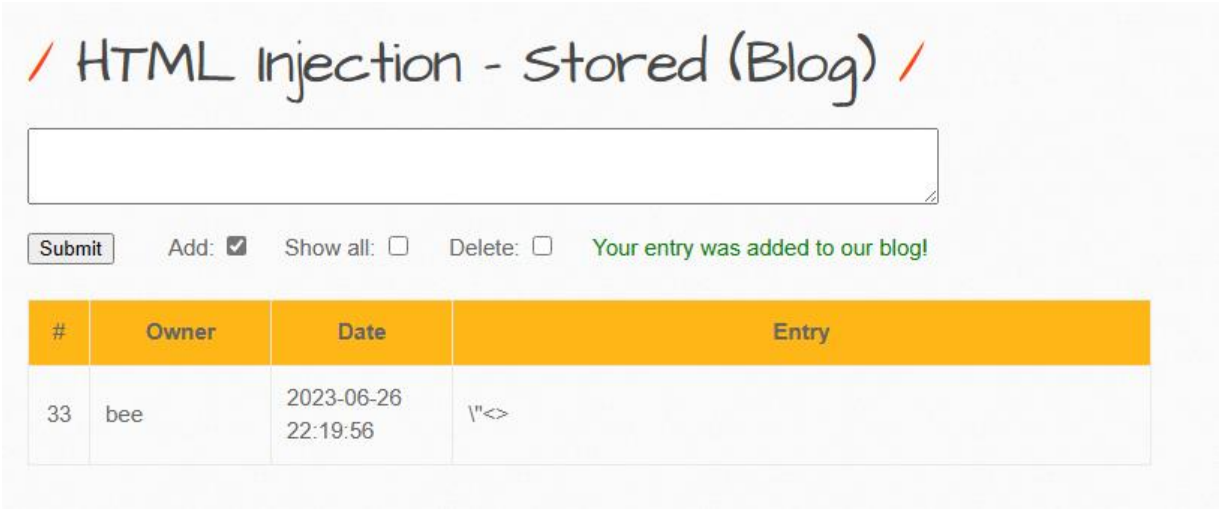
```
function xss_check_4($data){
// addslashes - returns a string with backslashes before characters that need to be quoted in database queries etc.
// These characters are single quote ('), double quote ("), backslash (\) and NUL (the NULL byte).
// Do NOT use this for XSS or HTML validations!!!
   return addslashes($data);
  }
```

Ở đây ta thấy được web đã sử dụng hàm addslashes. Hàm này có tác dụng cho thêm kí tự “\” vào trước những kí tự đặc biệt “”, “”, “NULL”

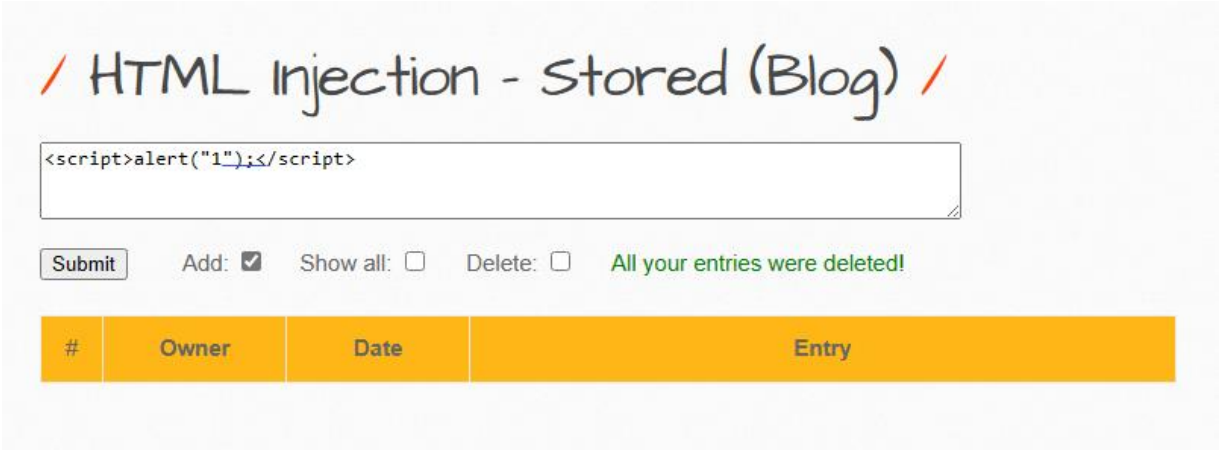
Giờ ta sẽ test thử filter của trang web



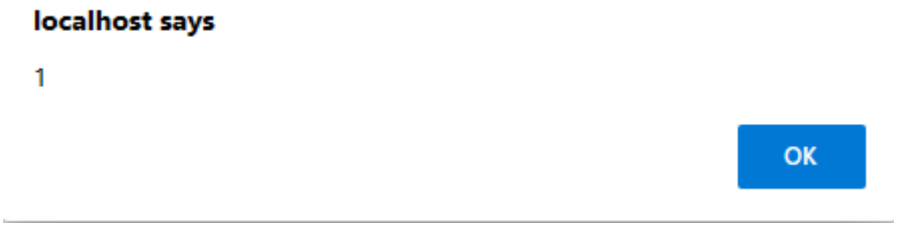
Và đây là kết quả khi chúng ta nhận được



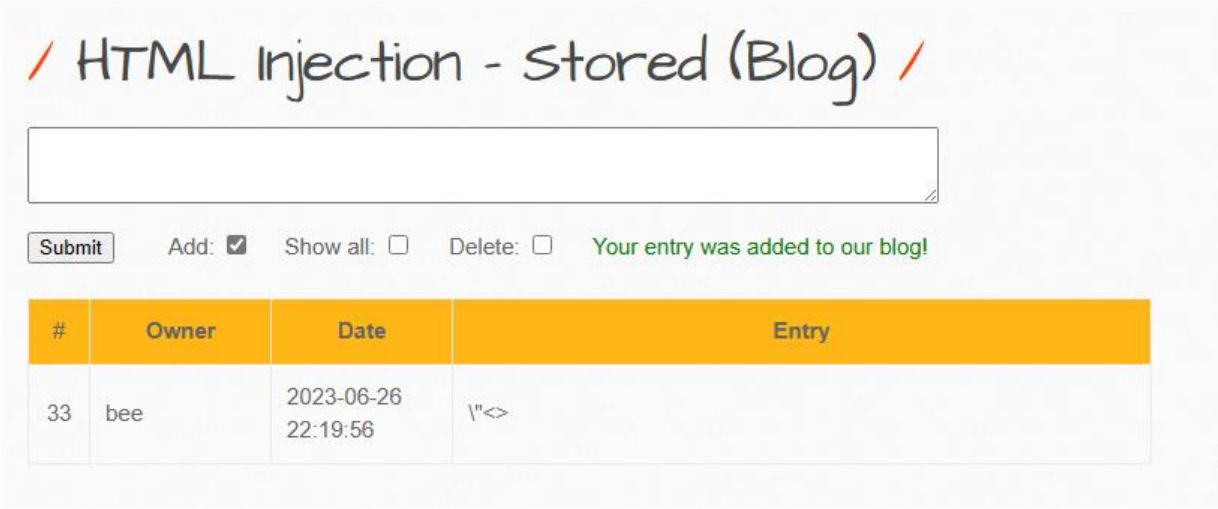
Ở đây ta thấy được filter của trang web đang bị lỗi và chỉ lọc các kí tự “”, “”” , và những kí tự này sẽ có trong một vài câu lệnh HTML chúng ta tiêm vào như là alert("1");. Vậy ta sẽ ví dụ nó với level 1 để xem alert() sẽ được thực hiện như nào



Và đây là kết quả chúng ta nhận được



Nhưng khi chúng ta chuyển sang level Medium thì câu lệnh chúng ta tiêm vào đã không còn tác dụng nữa. Chúng ta quay lại với những gì chúng ta đang làm với level Medium

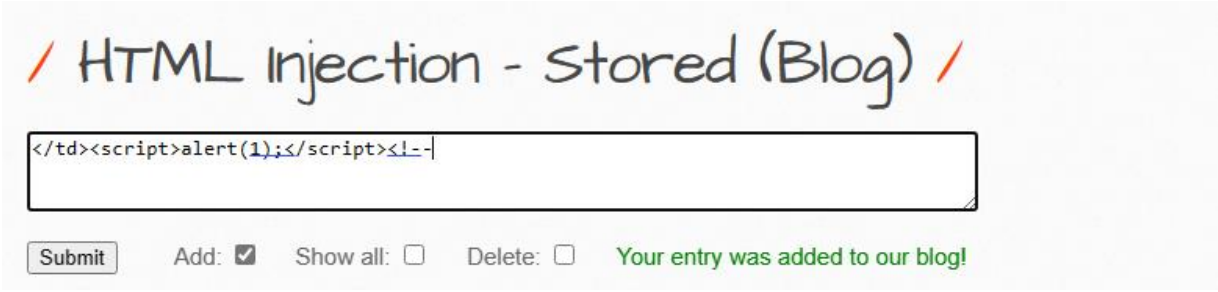


Chúng ta sẽ xem code HTML của những gì chúng ta đã tiêm vào

```
<td>\”<></td>
```

Giờ thì chúng ta sẽ tiêm một đoạn mã HTML để đánh lừa filter này

```
</td><script>alert(1)</script><!--
```



Và đây là kết quả ta nhận được

Trong đó </td>dùng để đóng thẻ HTML để toàn bộ đoạn code <script>alert(1)</script><!-- có thể được tiêm vào trang web mà không bị chuyển thành thực thể HTML, còn <!--là dùng để loại bỏ đoạn thừa ở đằng sau đi và đây là đoạn code sau khi chúng ta tiêm vào

```
<td></td>
<script>alert(1)</script>
<!--</td>
```

Vậy là ta đã thành công để đi qua level Medium

- Level High

Đối với level High trang web đã sử dụng hàm check xss_check_3

```
function xss_check_3($data, $encoding = "UTF-8"){
    // htmlspecialchars - converts special characters to HTML entities
    // '&' (ampersand) becomes '&amp;'
    // '"' (double quote) becomes '&quot;' when ENT_NOQUOTES is not set
    // "'" (single quote) becomes '&#039;' (or &apos;) only when ENT_QUOTES is set
    // '<' (less than) becomes '&lt;'
    // '>' (greater than) becomes '&gt;'
    return htmlspecialchars($data, ENT_QUOTES, $encoding);
}
```

Với hàm htmlspecialchars có tác dụng chuyển đổi toàn bộ những gì ta nhập vào thành thực thể HTML vì vậy chúng ta không thể tấn công một cách bình thường được. Nhưng nếu bạn ở trong hệ thông của trang web và bạn có thể vào được file .htaccess và thêm dòng mã AddDefaultCharest UTF-7 và thì trang web sec được chuyển sang mã hóa UTF-7 vì những kí tự “>”, “<”, “’” có điểm mã khác với UTF – 8 nên có thể đi qua filter của trang web