**1/ Báo cáo: Cross-Site Scripting (XSS) và Ảnh Hưởng của Nó**

Đây sẽ là một sự giới thiệu ngắn về cross-site scripting, loại lỗ hổng bảo mật được biểu diễn với từ viết tắt XSS.

Vậy cross-site scripting là gì? Nó đơn giản là khả năng thực thi mã JavaScript tùy ý trên trình duyệt của người dùng và hậu quả của việc đó là chúng ta kỹ có quyền truy cập vào phiên của người dùng và tất cả thông tin và xác thực liên quan đến điều đó. Thông thường, chúng ta thấy mức độ nghiêm trọng của một cuộc tấn công cross-site scripting trực tiếp tương quan với loại quyền mà một người dùng có.

Ví dụ cực đoan sẽ là trong bối cảnh của một ứng dụng ngân hàng chẳng hạn. Nếu đó chỉ là một trang web kiểu sách quảng cáo thông thường mà không có phiên hoặc đăng nhập người dùng, vẫn có một số cuộc tấn công có thể được tiến hành qua cross-site scripting nhưng mức độ nghiêm trọng có thể thấp hơn nhiều trong trường hợp khác.

Ở đây trong phòng thí nghiệm này, chúng ta có một ví dụ về cross-site scripting. Mục tiêu là để giải quyết bài này, thực hiện một cuộc tấn công cross-site scripting gọi hàm alert.

Đây là một loại reflected cross-site scripting và điều này có nghĩa là nó sẽ phản ánh cuộc tấn công trực tiếp dựa trên yêu cầu http mới nhất. Điều này khác biệt với stored cross-site scripting, nơi có thể lưu trữ cuộc tấn công cross-site scripting trong cơ sở dữ liệu và sau đó được khởi chạy sau này khi một người dùng truy cập vào một trang nhất định.

Ở đây cuộc tấn công sẽ được phản ánh trực tiếp như một kết quả của yêu cầu http chúng ta gửi đến ứng dụng. Hãy chú ý trên trang blog này, chúng ta có lựa chọn tìm kiếm blog và nếu chúng ta chỉ cần nhập một thuật ngữ tìm kiếm, chúng ta sẽ gọi nó là Ha Hien và chọn tùy chọn tìm kiếm. Hãy chú ý rằng thuật ngữ tìm kiếm chính xác mà chúng ta nhập vào ô tìm kiếm được phản ánh trên trang là Ha Hien. Bây giờ, trong khi ứng dụng nên xử lý điều này, trước hết khi chúng ta nhập thuật ngữ tìm kiếm của mình, phần backend phải đảm bảo rằng nó không cho phép một số ký tự đặc biệt như các thẻ html ví dụ như thẻ script và cách khác để giảm thiểu loại tấn công này là khi dữ liệu được trả về trang, nó nên được mã hóa sao cho các thẻ đặc biệt này sẽ không được truyền qua trình duyệt bằng cách nào đó, chúng sẽ chỉ được phản ánh trực tiếp vào trang.

Bây giờ chúng ta đã đặt một số thẻ h1 ở đó, hãy chỉ xem nguồn trang thực sự, thực tế chúng ta sẽ kiểm tra phần tử đó trực tiếp và chú ý rằng thẻ h1 đã được tiêm vào trang. Đó là lý do tại sao Ha Hien xuất hiện trên một dòng thứ hai vì h1 là một phần tử cấp block.

Đây không phải là cuộc tấn công thực sự vì ý tưởng ở đây là thực thi một đoạn mã JavaScript trong trường hợp này, nhưng điều này chỉ để chỉ ra rằng trang không xử lý các thẻ đặc biệt này đúng cách. Chúng ta không nên có các thẻ h1 được trả lại như một phần của html, chúng ta thực sự nên thấy thẻ h1 trên trang hoặc có thể ô tìm kiếm không chấp nhận các thẻ h1 hoặc thẻ html.

Vì vậy ta có thể thấy có một vấn đề rõ ràng với trang này. Cho dù với việc không xử lý các ký tự đó đúng cách, chúng ta có thể cố gắng nhập một thẻ script. Vì trong ngữ cảnh của một tài liệu html này theo mặc định sẽ tham chiếu đến JavaScript. Mục tiêu của bài này là sử dụng hàm JavaScript cụ thể là alert, mà chỉ cần mở một hộp thoại cảnh báo trên trang.

Hãy tìm kiếm và ta sẽ thấy hộp thoại cảnh báo hiển thị, điều này có nghĩa là thay vì xử lý các thẻ script đúng cách, nó thực sự tiêm chúng trực tiếp vào tài liệu và trình duyệt bây giờ đang thực thi JavaScript đó.

Đôi khi có thể mất một chút thời gian để hiểu rõ hậu quả của việc này, vì ta đã làm gì ở đây thực sự? Chúng ta chỉ mở một hộp thoại cảnh báo cho bản thân chúng ta. Vậy điều này sẽ hoạt động như thế nào trong ngữ cảnh của một cuộc tấn công?

Ý tưởng là ta có thể bao gồm điều này như một phần của URL. Nếu bạn nhận thấy trong thanh tìm kiếm ở trên, chúng ta thực sự có tham số tìm kiếm này và chúng ta có thể gửi một liên kết như vậy cho người dùng và nếu họ không thực sự hiểu rõ họ đang nhìn vào điều gì, nếu họ chỉ cần nhấp vào liên kết ví dụ như vậy, bây giờ chúng ta đang thực thi JavaScript trên trình duyệt của người dùng và rõ ràng chúng ta sẽ không chỉ mở một hộp thoại cảnh báo, chúng ta sẽ cố gắng tiến hành một loại tấn công bằng JavaScript. Và đây là một phần của URL, ta thấy chúng ta có kết quả tương tự. Vậy nếu một người dùng nhấp vào URL đó, họ cũng sẽ nhận được JavaScript này được thực thi trên trình duyệt của họ.

* Chúng ta đã giải quyết bài này cụ thể này là một cuộc tấn công reflected cross-site scripting.

**3/ Lab:**[**DOM XSS**](https://portswigger.net/web-security/cross-site-scripting/dom-based)**in document.write sink using source location.search**

Chúng ta sẽ giải đáp câu hỏi về DOM-based XSS và cụ thể là nó khác gì so với Reflected XSS. Vậy nên chúng ta sẽ sử dụng phòng thí nghiệm này và hướng dẫn như sau để giải quyết bài thí nghiệm này là thực hiện một cuộc tấn công cross-site scripting gọi hàm alert.

Khi khởi động phòng thí nghiệm. Đây là một loại blog và chúng ta có một đầu vào biểu mẫu ở đây nơi chúng ta có thể tìm kiếm blog. Vậy hãy tìm kiếm một cái gì đó sẽ là duy nhất và dễ tìm thấy khi chúng ta duyệt qua mô hình đối tượng tài liệu với firefox, chúng ta có thể kéo lên bảng phát triển với tổ hợp phím control, shift và i và tab kiểm tra này là trình duyệt DOM, vì vậy điều chúng ta đang nhìn thấy ở đây là mô hình đối tượng tài liệu.

Bây giờ nếu chúng ta quen thuộc với các ngôn ngữ lập trình như javascript chẳng hạn, ta có thể đã hiểu được về một đối tượng là gì. Bây giờ mô hình đối tượng tài liệu không phải là javascript nhưng nó có thể được điều chỉnh bởi javascript.

Một trang được thiết lập để điều chỉnh DOM theo một cách nhất định và thông qua đầu vào của chúng ta, chúng ta có thể làm hỏng cách mà javascript đang điều chỉnh DOM. Bây giờ chúng ta đã có một cuộc tấn công cross-site scripting dựa trên DOM.

Đầu ra trang cụ thể này hữu ích để hiểu về vấn đề này vì nó thực sự hiển thị cụm từ tìm kiếm của chúng ta hai lần như một phần của mô hình đối tượng tài liệu. Chúng ta chỉ cần tìm kiếm z3nsh3ll và ta sẽ thấy nó xuất hiện hai lần như một phần của mô hình đối tượng tài liệu.

Sự xuất hiện đầu tiên chúng ta thấy bên trong một thẻ h1 và điều này rất có thể đang được trả lại như một phần của phản hồi HTTP trực tiếp từ trang. Ví dụ nếu chúng ta có thể tiêm một thẻ script ở đây, điều đó thực sự là một phần của phản hồi HTTP từ trang vì vậy đó là một cuộc tấn công cross-site scripting được phản ánh. Hãy xem xét sự xuất hiện thứ hai của cụm từ tìm kiếm này trong DOM và chúng ta có thể thấy nó ở bên trong một thẻ hình ảnh trước khi đóng thẻ cụ thể này.

Vậy làm thế nào nó đến đó, hãy chú ý ngay phía trên chúng ta có một thẻ script này là javascript.

Thì javascript này đang làm gì? Vậy nó đang xác định một hàm, hàm được gọi là track search và nó nhận một tham số là truy vấn. Hàm này gọi document.write, document.write là một phần của cách mà javascript điều chỉnh DOM, nó sẽ ghi bất kỳ thứ gì đang trong tham số đó cho hàm document write vào DOM. Vậy nó đang ghi gì, nó đang ghi một thẻ nguồn hình ảnh có vẻ quen thuộc chúng ta vừa nhìn thấy nó được hiển thị trong DOM nguồn hình ảnh tài nguyên hình ảnh theo dõi. gif dấu hỏi

vì vậy chúng ta đang đi vào chuỗi truy vấn bây giờ các thuật ngữ tìm kiếm equals truy vấn vậy truy vấn này từ đâu, biến truy vấn mới có lớp tìm kiếm URL này là một lớp javascript tích hợp và nó gọi hàm tạo với tham số window.location. search. window.location.search là bất cứ điều gì sau dấu hỏi trong thanh url ở đầu trang vì vậy hiện tại nó chỉ đọc search equals z3nsh3ll vì vậy url search params sẽ lấy chuỗi đó search equals z3nsh3ll và nó đã mã hóa sẵn với logic để phân tách nó thành các cặp khóa giá trị chúng ta sau đó có thể lấy đối tượng url search params javascript mới được xây dựng này và sau đó gọi phương thức .get trên nó mà chúng ta thấy đang diễn ra tiếp theo và thực ra điều đó đang yêu cầu đối tượng làm gì là tìm giá trị có khóa tìm kiếm và trong trường hợp này kết quả sẽ là z3nsh3ll. Vậy kết quả cuối cùng ở đây là biến query được gán giá trị là chuỗi z3nsh3ll.

Sau đó chúng ta có một điều kiện, nếu có truy vấn và chúng ta biết có một truy vấn vì nó vừa được gán, nó sẽ gọi hàm đã định nghĩa là track search và nó sẽ truyền tham số truy vấn là chuỗi z3nsh3ll trong trường hợp này. Vì vậy nó chắc chắn sẽ được gọi và nó sẽ ghi thẻ hình ảnh đó vào DOM và nó sẽ bao gồm z3nsh3ll làm tham số của các thuộc tính tìm kiếm của hình ảnh đó.

A screen shot of a computer program

Description automatically generated

Đó là lý do tại sao chúng ta có image source equals resources images tracker.gif sau đó chúng ta có dấu hỏi chỉ ra đầu của chuỗi truy vấn và chúng ta có search terms equals z3nsh3ll. Vậy chúng ta có thể thấy cả hai trường hợp của cụm từ tìm kiếm z3nsh3ll đã hiển thị dưới dạng của trang theo các cách khác nhau.

Trường hợp đầu tiên của z3nsh3ll đã được gửi như một phần của phản hồi HTTP trực tiếp đến yêu cầu của chúng ta. Trường hợp thứ hai kết thúc một cách trực tiếp vì nó phải được bao gồm hoặc được viết vào DOM bằng cách sử dụng hàm javascript document.write này. Vì vậy nếu chúng ta tấn công phản hồi như một phần của phản hồi HTTP trực tiếp thì đó sẽ là cuộc tấn công cross-site scripting được phản ánh. Nếu chúng ta tấn công cách mà javascript này đang điều chỉnh DOM thì đó sẽ là một cuộc tấn công cross-site scripting dựa trên DOM.

Vậy hãy thực hiện ngay bây giờ. Bước đầu tiên trong điều này là xem xét bối cảnh nơi z3nsh3ll được hiển thị trong DOM. Vậy nó là một phần của thuộc tính nguồn của thẻ hình ảnh. Bây giờ nếu chúng ta có thể thoát khỏi thuộc tính đó và cho thẻ hình ảnh một thuộc tính thứ hai ví dụ như bỏ tải bằng cách báo động để một khi hình ảnh đã tải xong thì nó chạy hàm alert.

Vậy hãy xem liệu chúng ta có thể làm được điều đó không. Vậy chúng ta sẽ nhập cụm từ tìm kiếm của mình, chúng ta sẽ bao gồm một dấu ngoặc kép để thoát khỏi thuộc tính cụ thể đó, sau đó chúng ta sẽ bắt đầu định nghĩa một thuộc tính mới onload bằng cách gọi hàm javascript alert mà đó là mục tiêu của bài này và nếu ta nghĩ về điều đó thì dấu ngoặc kép cuối cùng đã được bao gồm bởi vì nó ban đầu bọc hoặc kết thúc thẻ nguồn đó vì vậy chúng ta thực sự có thể xóa dấu ngoặc kép cuối cùng.

Hãy tìm kiếm điều này và xem điều gì xảy ra và chú ý chúng ta nhận được cửa sổ cảnh báo trên trang cho thấy một cuộc tấn công cross-site scripting thành công. Vậy một chút phân tích sau đây, hãy xem điều gì đã xảy ra với DOM. Vậy nếu chúng ta kiểm tra DOM và xem điều gì đã xảy ra ở đây, chúng ta có image source equals nó kết thúc với z3nsh3ll sau đó chúng ta có dấu ngoặc kép cuối cùng và sau đó nhìn vào điều này chúng ta đã tiêm một thuộc tính bổ sung vào hình ảnh đó thông qua hàm document.write javascript.

Vậy điều này không phải là một phần của phản hồi HTTP ban đầu đến trang, điều này thực sự là một thay đổi đã được thực hiện trên DOM dựa trên việc rằng javascript thực thi một khi trang đã tải xong và chúng ta có thể chứng minh điều đó nếu chúng ta nhìn vào mã nguồn trang và tìm kiếm z3nsh3ll chúng ta có thể thấy sự xuất hiện ban đầu của z3nsh3ll trong thẻ h1 đó là một phần của phản hồi HTTP trực tiếp từ máy chủ. Chúng ta thậm chí có javascript ở đây nhưng chúng ta không thấy sự xuất hiện thứ hai của z3nsh3ll như một phần của trang vì xem mã nguồn không hiển thị các thao tác DOM, nó chỉ hiển thị html gốc đã được gửi từ máy chủ. Các thay đổi được theo dõi trong DOM nhưng chúng không được phản ánh vào html của trang khi chúng ta xem mã nguồn.

**5/ DOM XSS in jQuery anchor href attribute sink using location.search source**

Chúng ta sẽ tiến hành một cuộc tấn công Cross-Site Scripting nhưng sẽ sử dụng một jQuery sink và điều này thực sự chỉ để cho các ta thấy rằng có một loạt các sink khả dụng cho một cuộc tấn công Cross-Site Scripting và không nhất thiết phải sử dụng JavaScript thuần túy.

Nếu không biết về jQuery, nó khá tương tự như một framework JavaScript nhưng nó lại là một thư viện hỗ trợ hơn, không được coi là một framework đầy đủ nhưng nó cho phép chúng ta thao tác với DOM mà không cần sử dụng JavaScript verbose và ta không nói rằng có gì sai trái với JavaScript thuần túy nhưng đôi khi các nhà phát triển có thể muốn viết ít dòng mã hơn.

Vậy là chúng ta sẽ sử dụng một lab DOM Cross-Site Scripting trong jQuery anchor href attribute sync sử dụng location.search source, sau đó chúng ta được cung cấp một số hướng dẫn nói rằng để giải quyết lab này hãy hiển thị ngược lại cookie tài liệu.

xét lab này. Bây giờ chúng ta biết có một liên kết ngược lại dễ bị tấn công, vì vậy bước đầu tiên ở đây thực sự là xác định nơi mà liên kết ngược lại dễ bị tấn công đó nằm ở đâu, có thể mất một vài phút để kích vào trang để tìm ra điều chúng ta đang tìm kiếm, thử chuyển đến phần gửi phản hồi, chúng ta có thể gửi biểu mẫu phản hồi nhưng chúng ta cũng có một tùy chọn để quay lại trang trước, nhưng lưu ý rằng đó đã được mã hóa như là một phần của trang, nó không phải là liên kết ngược lại trong trình duyệt mà chúng ta có thể thay đổi chức năng của nó, ta sẽ kiểm tra liên kết ngược lại đó,

nó là một thẻ mô phỏng hoặc liên kết, có id là backlink và sau đó nó có thuộc tính href, vì vậy đây là nơi mà liên kết sẽ dẫn chúng ta và chúng ta có thể thấy nếu chúng ta kích vào đó chúng ta sẽ chỉ được dẫn đến dấu gạch chéo - thư mục gốc của trang web, câu hỏi là làm thế nào mà thuộc tính href đó lại ở đó. Tất nhiên đôi khi nó có thể là một phần của phản hồi ban đầu HTTP từ cuộc khảo sát và trong trường hợp đó thực sự không có nhiều chúng ta có thể làm để thay đổi nó ít nhất không nếu chúng ta muốn thực thi mã JavaScript tùy ý nhưng nếu chúng ta kiểm tra thẻ script này chúng ta sẽ thấy rằng phần tử đó thực sự được đặt ở đó bởi jQuery mà đã thay đổi DOM sau khi nhận phản hồi HTTP từ máy chủ vì vậy chúng ta có một hàm được xác định ở đó và có lẽ đang được thực thi ngay lập tức và ta có thể thấy chúng ta có dấu gạch chéo nên dấu này là gì, nó không trông giống như JavaScript, điều này là bởi vì nó là một phần của thư viện jQuery và dấu gạch chéo này là một bộ chọn giúp chúng ta chọn một số phần tử nhất định nếu bạn đã làm việc với CSS bạn sẽ quen với ý tưởng này vì vậy khi chúng ta có bộ chọn dấu gạch chéo sau đó trong ngoặc đơn chúng ta có một tham số với dấu gạch chéo số dấu thăng với id backlink nếu quen với CSS ta sẽ biết rằng điều này đang chọn phần tử với id backlink

Đúng đó là phần tử chúng ta vừa nhìn thấy có thuộc tính href. Sau đó nó gọi một hàm jQuery attr mà thực sự sẽ gán thuộc tính href đó cho giá trị được cung cấp trong tham số thứ hai ở đây. Vậy giá trị được cung cấp hiện tại là gì? Đó là new url search params là một lớp JavaScript mà được khởi tạo ở đây thành một đối tượng và nó lấy vào tham số window.location.search vì vậy đó là mọi thứ trong thanh URL sau dấu hỏi vì vậy vào lúc này nó đọc return path equals forward slash vì vậy đó cũng là một gợi ý lớn nữa, sau đó nó lấy giá trị của khóa return path vì vậy chúng ta vừa thấy nó là một dấu gạch chéo trong mọi trường hợp, nó làm gì vậy, chúng ta đang nhìn thấy rằng nó đang nhìn vào URL, nó đang tìm kiếm tham số với khóa return path, nó lấy giá trị vì vậy là dấu gạch chéo trong trường hợp này, sau đó nó thực thi DOM sao cho thẻ anchor mà chúng ta đã xem có giá trị href được đặt thành giá trị đó đang được truyền dưới khóa return path trong chuỗi truy vấn vì vậy dấu gạch chéo trong trường hợp này, điều chúng ta thực sự thấy là giá trị href đang được xác định trực tiếp bởi tham số đang được truyền như một phần của chuỗi truy vấn dưới khóa return path vì vậy chúng ta thực sự có thể thao tác nó và nó có nghĩa là chúng ta có thể kiểm soát giá trị của thuộc tính href đó vậy câu hỏi là chúng ta muốn thiết lập nó thành cái gì, trong các cuộc tấn công Cross-Site Scripting trước đó chúng ta đã làm những điều như thêm các thẻ script chúng ta đã làm những điều như thêm một thẻ hình ảnh với nguồn là số không và sau đó khi điều đó gây ra lỗi chúng ta gọi hàm alert nhưng quan trọng là phải xem xét sink và ngữ cảnh nơi chúng ta chèn vào vì vậy bạn có thể thấy ngữ cảnh ở đây thực sự là chúng ta đang ở bên trong thuộc tính href và chúng ta không thực sự sẽ có khả năng thoát ra khỏi đó vì cách mà jQuery hoạt động vậy câu hỏi là liệu chúng ta có thể thực thi JavaScript từ bên trong một thuộc tính href

và câu trả lời thực sự là có, Nếu chúng ta có một thẻ mô phỏng và bên trong href chúng ta có javascript theo sau là một số mã JavaScript thì JavaScript đó sẽ được thực thi trực tiếp vì vậy bình thường cách chúng ta nghĩ về các thẻ mô phỏng là nếu người dùng kích vào thẻ mô phỏng thì trình duyệt sẽ chuyển hướng đến URL chứa trong thuộc tính href nhưng điều gì sẽ xảy ra trong trường hợp này là thay vì truy cập vào một URL khác khi người dùng kích vào thẻ mô phỏng nó sẽ thực thi trực tiếp mã JavaScript được chứa trong thuộc tính href .

**7/ Reflected XSS into attribute with angle brackets HTML-encoded**

Vậy chúng ta sẽ xem xét một cuộc tấn công cross-site scripting phản ánh và nó sẽ tận dụng một đầu vào tìm kiếm có lỗi nên hãy xem xét lab. Vì vậy, chúng ta có một trang dạng Blog và chúng ta có tùy chọn để tìm kiếm blog và một trong những điều đầu tiên mà chúng ta thích làm khi tìm kiếm các vector tấn công cross-site scripting là nhập một chuỗi tùy ý và chỉ xem nó xuất hiện ở đâu trên trang nên hãy tìm kiếm một chuỗi, hãy mở công cụ phát triển và hãy tìm kiếm chuỗi tùy ý đó trong Dom, chúng ta có hai kết quả, kết quả đầu tiên nằm bên trong thẻ H1, khả năng cao là chúng ta sẽ không thể làm gì đó với điều đó, vì vậy hãy xem kết quả xuất hiện thứ hai của chuỗi tùy ý của chúng ta, vì vậy chúng ta có một trường nhập loại bằng văn bản giữ chỗ tìm kiếm Tên Blog bằng tìm kiếm giá trị bằng Zen shell, vậy nghĩ về ngữ cảnh ở đây là thuật ngữ tìm kiếm của chúng ta xuất hiện bên trong thuộc tính giá trị của phần tử đầu vào đó, vậy câu hỏi là chúng ta có thể phá vỡ khỏi thuộc tính giá trị đó và tạo ra một thuộc tính mới ví dụ như một sự kiện on Mouse over và sau đó chúng ta có thể hiển thị một hộp thoại cảnh báo JavaScript khi sự kiện đó xảy ra, vì vậy cách chúng ta sẽ phá vỡ khỏi đó là bằng cách kết thúc với dấu ngoặc kép vậy hy vọng rằng chúng ta ở bên ngoài thuộc tính giá trị đó bây giờ nhưng vẫn ở bên trong các thẻ đầu vào nên hãy xem xem chúng ta có thể tạo ra một thuộc tính mới, đó sẽ là một sự kiện on Mouse over và khi sự kiện đó xảy ra, tức là khi con trỏ của người dùng nằm trên phần tử cần xem xét, chúng ta muốn gọi hàm cảnh báo JavaScript vậy hãy tìm kiếm, bạn có thể thấy URL ở đầu trang được cập nhật vì tất cả thông tin này được chứa trong chuỗi truy vấn, vậy nên chúng ta có tham số tìm kiếm đó và giá trị là phiên bản đã được mã hóa URL của chuỗi tìm kiếm của chúng ta, bây giờ bạn có thể thấy ngay lập tức chúng ta nhận được lời chúc mừng bạn đã giải quyết được lab Echo vào trang lưu ý điều gì xảy ra nếu chúng ta di chuột qua ô tìm kiếm đó chúng ta nhận được cảnh báo đúng như một chút phân tích sau khi xem xét xem chúng ta có thể kiểm tra phần tử đó trong thực tế rất khó để nhấp chuột phải vào phần tử trực tiếp vì nó rõ ràng kích hoạt cảnh báo vậy nhìn vào Dom ở đây chúng ta có một biểu mẫu chúng ta có phần tử đầu vào của chúng ta tiếp tục từ nơi chúng ta dừng lại chúng ta có giá trị bằng Zen shell và sau đó chúng ta có thuộc tính on Mouse over bây giờ bạn có thể nhận ra rằng kết quả cuối cùng của điều này sẽ là một dấu ngoặc kép dư thừa mà chúng ta thực sự có thể thấy ở đó trong Dom điều này không dường như là HTML hợp lệ nhưng Dom khá thông minh và đôi khi vẫn có thể hiểu được điều được dự định bởi một phần tử cụ thể ngay cả khi nó không khớp hoàn toàn với cú pháp đúng vậy nó không dường như là cú pháp đúng đối với tôi nhưng bạn vẫn có thể thấy rằng phần còn lại của phần tử vẫn hoạt động tốt, bây giờ bạn có thể tự hỏi là chúng ta làm thế nào để làm được điều này không có vẻ như chúng ta nên có thể phá vỡ khỏi một thuộc tính trong một phần tử và nó không phải là một lỗ hổng với HTML hoặc thậm chí là một lỗ hổng với Dom nó chỉ là một lỗ hổng với cách mà Blog cụ thể này đã được mã hóa bởi nhà phát triển và chúng ta ph