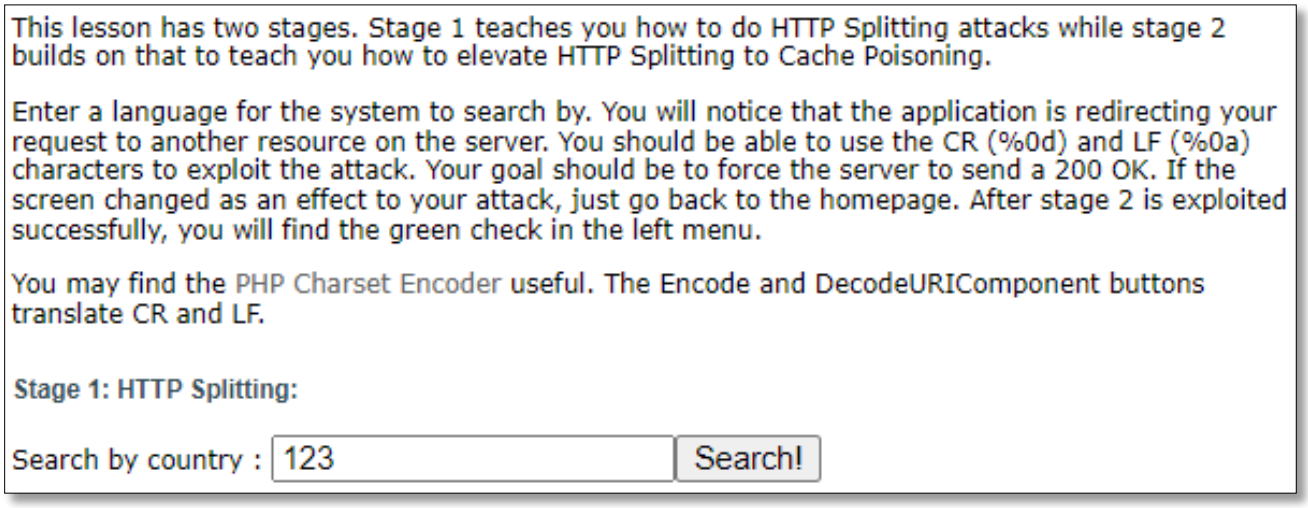
General - HTTP Splitting

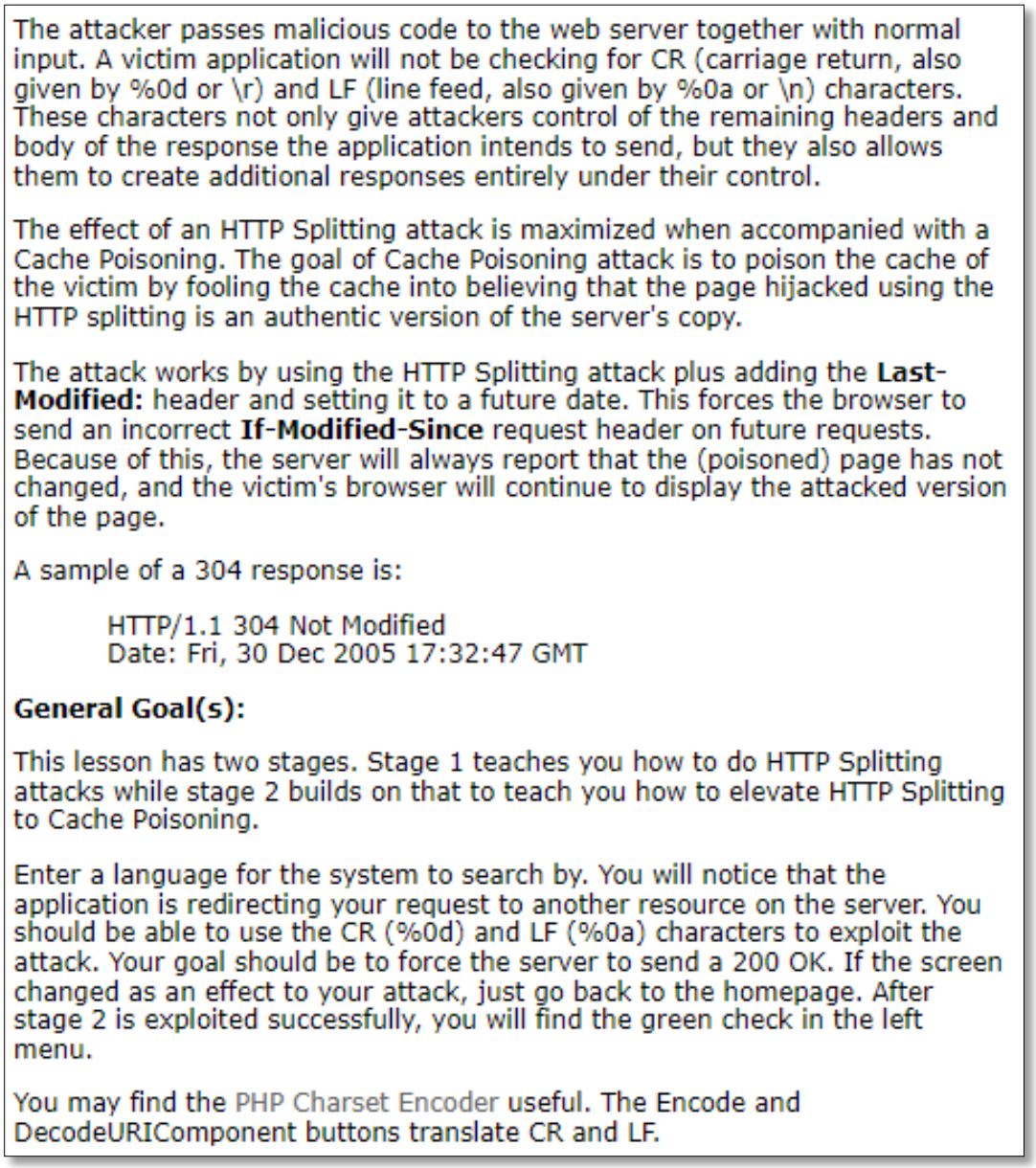
Tấn công HTTP Response Splitting nhằm vào việc các dữ liệu đi vào ứng dụng web thông qua một nguồn không đáng tin cậy, khiến cho người dùng có thể gửi đi 1 request HTTP chứa các ký tự độc hại (ký tự xuống dòng \r - Carriage Return và \n - Line Feed), và nội dung yêu cầu đó được hiển thị ngược trở lại trong response của Server.

Kẻ tấn công có thể lợi dụng các ký tự xuống dòng được phía Server cho phép mà không lọc để tạo ra các tiêu đề phản hồi bổ sung giả mạo trong gói response của gói tin HTTP, nhằm kết thúc phản hồi đầu tiên do Server trả về và tạo ra 1 phản hồi khác chứa mục đích của cuộc tấn công.



Trong bài tập này đưa ra 2 yêu cầu, bước thứ nhất là thực hiện thành công 1 cuộc tấn công HTTP Splitting, yêu cầu còn lại là dựa trên việc thành công đó, thực hiện việc Cache Poisoning (thông thường thì các request tới WebServer sẽ được Cache tại đâu đó "Browser, Router, Proxy, …", và nếu các bản Cache này được sử dụng chung cho nhiều người, khi ta thực hiện đầu độc các bản Cache này, phản hồi được lưu trong bộ nhớ Cache thì những người dùng đó sẽ tiếp tục nhận nội dung

độc hại cho đến khi mục nhập bộ nhớ Cache bị xóa hoặc cập nhật, mặc dù chỉ người dùng của phiên bản trình duyệt cục bộ mới bị ảnh hưởng) bằng cách thêm các Header "Last-Modified:" bằng 1 thời điểm trong tương lai theo khung GMT.



The attacker passes malicious code to the web server together with normal input. A victim application will not be checking for CR (carriage return, also given by %0d or \r) and LF (line feed, also given by %0a or \n) characters. These characters not only give attackers control of the remaining headers and body of the response the application intends to send, but they also allows them to create additional responses entirely under their control.

The effect of an HTTP Splitting attack is maximized when accompanied with a Cache Poisoning. The goal of Cache Poisoning attack is to poison the cache of the victim by fooling the cache into believing that the page hijacked using the HTTP splitting is an authentic version of the server's copy.

The attack works by using the HTTP Splitting attack plus adding the **Last-Modified:** header and setting it to a future date. This forces the browser to send an incorrect **If-Modified-Since** request header on future requests. Because of this, the server will always report that the (poisoned) page has not changed, and the victim's browser will continue to display the attacked version of the page.

A sample of a 304 response is:

HTTP/1.1 304 Not Modified
Date: Fri, 30 Dec 2005 17:32:47 GMT

General Goal(s):

This lesson has two stages. Stage 1 teaches you how to do HTTP Splitting attacks while stage 2 builds on that to teach you how to elevate HTTP Splitting to Cache Poisoning.

Enter a language for the system to search by. You will notice that the application is redirecting your request to another resource on the server. You should be able to use the CR (%0d) and LF (%0a) characters to exploit the attack. Your goal should be to force the server to send a 200 OK. If the screen changed as an effect to your attack, just go back to the homepage. After stage 2 is exploited successfully, you will find the green check in the left menu.

You may find the PHP Charset Encoder useful. The Encode and DecodeURIComponent buttons translate CR and LF.

Giả sử bạn đã tạo một trang web và đặt tiêu đề "Last-Modified:". Khi người dùng duyệt trang web của bạn, trình duyệt của người dùng tạm thời lưu trữ (bộ đệm) một số thông tin như HTML, hình ảnh, biểu định kiểu và thậm chí một số nội dung động

không thường xuyên thay đổi (là cái "Last-Modified:" ở trên", chỉ ra ngày cuối cùng mà nội dung website được sửa đổi.).

Tiêu đề "If-Modified-Since" gửi yêu cầu đến máy chủ cùng với giá trị của "Last-Modified:" được lưu trữ trong Cache nhằm xem liệu trình duyệt có nên sử dụng bản sao được lưu trong bộ nhớ Cache hay tải xuống phiên bản mới hơn của trang web hay không. Và với việc tấn công vào "Last-Modified:" bằng 1 mốc thời gian trong tương lai, giá trị của "If-Modified-Since" mỗi khi gửi tới máy chủ sẽ là 1 giá trị không phù hợp, khiến Website sẽ không trả về kết quả cập nhật mới cho Cache Website, làm cho người dùng sẽ luôn sử dụng phiên bản bị tấn công của trang web trong bộ nhớ đệm.

Okay, lý thuyết lan man, ta sẽ đi vào thực nghiệm.

Đầu tiên là sử dụng chức năng và phân tích request - response 1 chút.

```
POST /WebGoat/lessons/General/redirect.jsp?Screen=3&menu=100 HTTP/1.1
Host: 192.168.48.131
Content-Length: 29
Cache-Control: max-age=0
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.48.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,i
mage/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer:
http://192.168.48.131/WebGoat/attack?Screen=3&menu=100&fromRedirect=yes&langu
age=123
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=C3F0EC859EA4D36F33D80BECC17FCBC2
Connection: close

language=123&SUBMIT=Search%21
```

Request gửi đi 1 tham số giá trị "language=123", đây cũng là vùng giá trị ta sẽ tấn công theo yêu cầu đề bài.

```
1  HTTP/1.1 302 Moved Temporarily
2  Date: Thu, 17 Nov 2022 01:45:08 GMT
3  Server: Apache-Coyote/1.1
4  Location:
   http://192.168.48.131/WebGoat/attack?Screen=3&menu=100&fromRedirect=yes&langu
   age=123
5  Content-Type: text/html;charset=ISO-8859-1
6  Via: 1.1 127.0.1.1
7  Vary: Accept-Encoding
8  Content-Length: 0
9  Connection: close
10
11
```

Và, whoala, cái tham số giá trị "123" kia được Server sử dụng lại trong phản hồi, suất hiện trong "Location". Ta có thể lợi dụng điều này để ngắt quãng HTTP Response ban đầu của Server và tạo ra 1 Response của riêng ta.

Hãy thử tạo 1 Respone đơn giản bằng việc hiển thị 1 nội dung HTML.

```
Content-Length: 0


Http/1.1 200 OK
Content-Type: text/html
Content-Length: 21
<html> Hacked </html>
```

Phía trên là một phản hồi cơ bản để hiển thị một nội dung HTML cho người dùng. Giờ, hãy kết hợp nó với phản hồi phía Server để phân tích 1 chút.

Phản hồi sẽ trông thế này:

HTTP/1.1 302 Moved Temporarily

Date: Thu, 17 Nov 2022 01:45:08 GMT

Server: Apache-Coyote/1.1

Location: http://192.168.48.131/WebGoat/attack?Screen=3&menu=100&fromRedirect=yes&language=

Content-Length: 0

Http/1.1 200 OK

Content-Type: text/html

Content-Length: 21

<html> Hacked </html>

Content-Type: text/html;charset=ISO-8859-1

Via: 1.1 127.0.1.1

Vary: Accept-Encoding

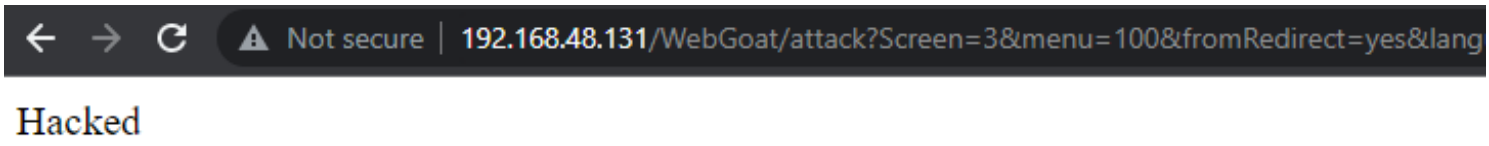
Content-Length: 0

Connection: close

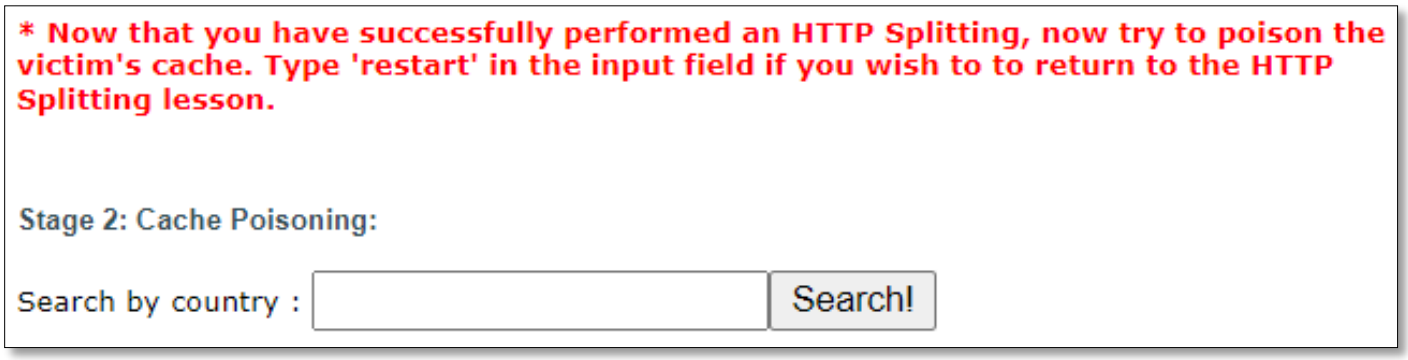
Lý do mà ta cần thêm "Content-Length: 0" là vì đây thường là tiêu đề kết thúc các Header HTTP, và giá trị của nó chỉ định số byte nội dung sẽ được phản hồi về cho người dùng (nội dung thường cách các tiêu đề 1 dòng trắng, hoặc không cũng được), "Content-Length: 0" sẽ chỉ định yêu cầu đầu tiên không có nội dung trả về, và tiếp tục thực hiện response thứ 2, với nội dung trả về là 21 byte "**<html> Hacked </html>**", giới hạn 21 byte để nó không lấy thêm bất cứ giá trị nào phía sau nữa.

Cái dòng trắng không có nội dung ở phía dưới "Content-Length: 0" chính là 0 byte nội dung nhé.

Ok, form của bài này không cho phép ta thêm ký tự xuống dòng vào, vậy thì giờ decode URL rồi gửi payload trên để tấn công thôi, **(Gói tin HTTP sử dụng decode URL, và decode URL có thể decode cả ký tự xuống dòng, nên phía Server có thể hiểu được việc xuống dòng này).**



Đúng như mong muốn, payload đã khiến Server phản hồi về lại chỉ một nội dung là "**<html> Hacked </html>**".



Giờ thì thực hiện bước thứ 2, Cache Poisoning. Như đã giải thích ở trên, những gì ta cần làm là thêm 1 Header HTTP "Last-Modified:" trong phản hồi của Server và buộc người dùng phải gửi lại yêu cầu với "If-Modified-Since" để cập nhật lại Cache website, nhưng với 1 thời gian ở tương lai, việc cập nhật này sẽ xảy ra lỗi (giải thích ở trên).

Lấy mẫu 1 câu "Last-Modified:" trên mạng về và sửa thời gian.

Last-Modified: Wed, 21 Oct 2055 07:28:00 GMT

Chèn thêm nó vào với payload ban nãy. Giờ thì ta không cần tới đoạn hiển thị HTML nữa đâu, nãy cần cho việc ktra payload thôi, các giá trị của HTTP response ban đầu bị ta cắt ngang sẽ được thêm vào ở dưới.

Content-Length: 0

Http/1.1 200 OK

Last-Modified: Wed, 21 Oct 2055 07:28:00 GMT