Author: Wong Tin Kit
Student ID: 1003331

Task 1: Observing HTTP Request.

In Fig 1.0, a GET Request made for the resource http://www.csrflabelgg.com/profile/samy. This request is made when a user is trying to view Samy's Profile Page. In this case, there are no parameters used in the URL Scope.

In Fig 1.1, a POST Request is made to the resource http://www.csrflabelgg.com/action/profile/edit . This request is made when Samy saves the changes made on her Profile Page. There are no parameters used in the URL Scope. The parameters associated with POST request are attached in the body of the Request as seen in the second box of the POST Request itself (eg. __elgg_token, __elg_ts, etc).

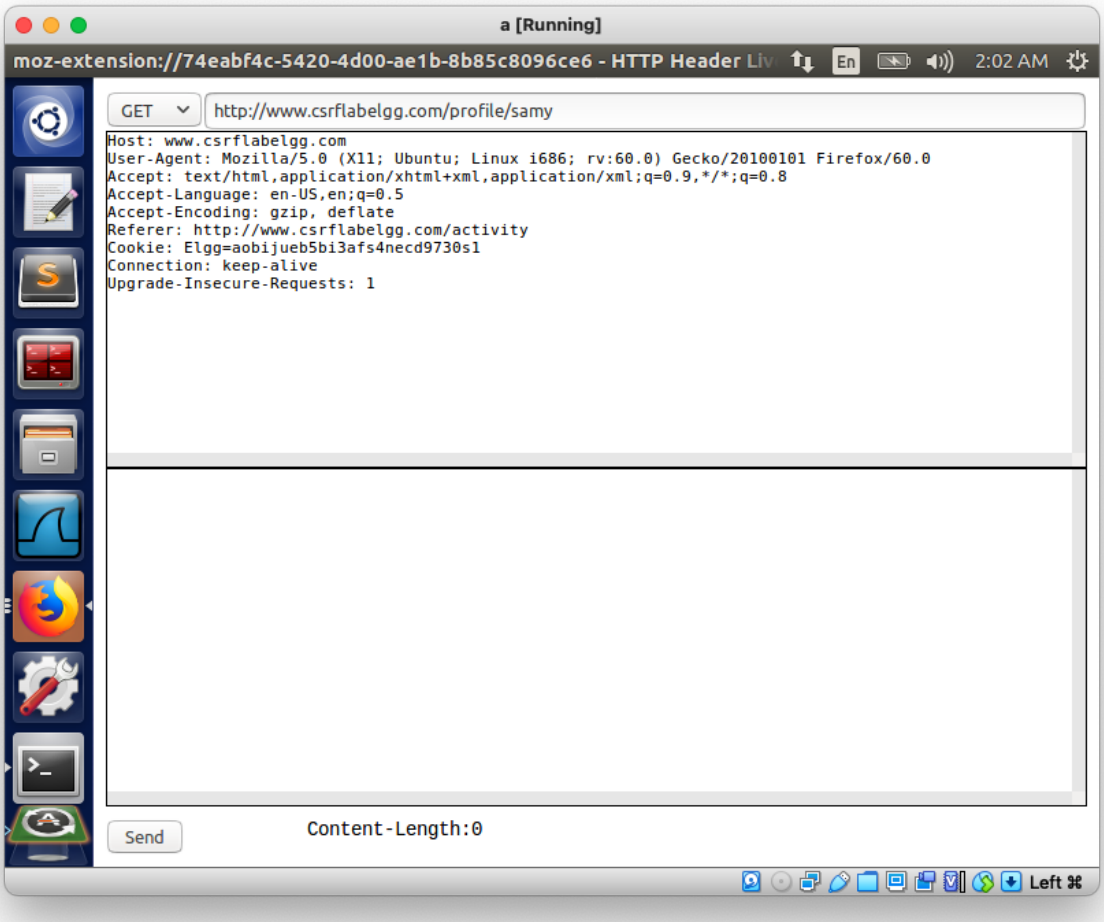

Fig. 1.0 GET Request

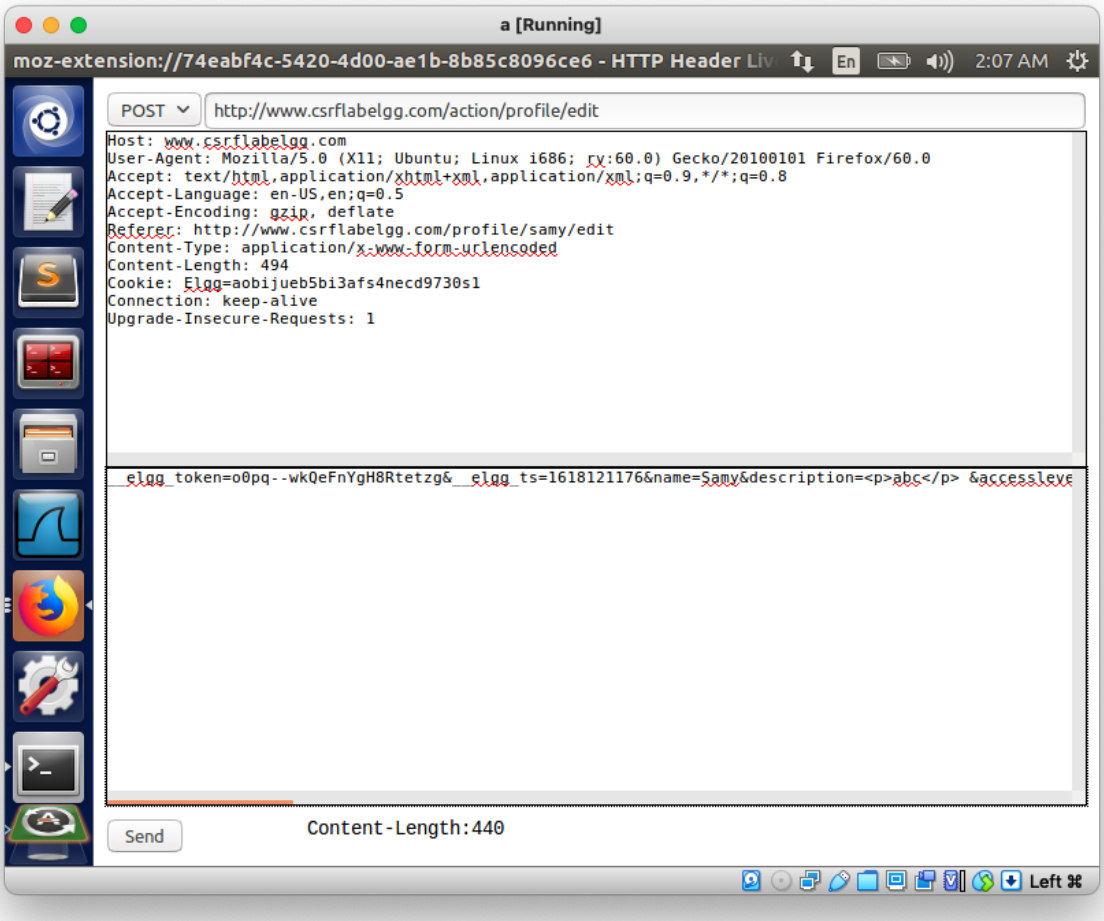

Fig. 1.1 POST Request

Task 2: CSRF Attack using GET Request

Pretend that you are Boby, describe how you can construct the content of the web page, so as soon as Alice visits the web page, Boby is added to the friend list of Alice (assuming Alice has an active session with Elgg).

We must first understand how Elgg's Add Friend Service works. After investigation (Fig. 2.0) , it sends a GET Request to the endpoint http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=1618124123&__elgg_token=GW8oVx3BbOevwsUr91koSw&__elgg_ts=1618124123&__elgg_token=GW8oVx3BbOevwsUr91koSw where 43 is the GUID of the friend to be added.

As Boby, I would construct the malicious web page content to be related to the email used in sending the link to this web page. Alice will be less suspicious of this malicious web page. In the content of the webpage, I will embed an image tag and map the src attribute to a GET Request for Boby's Elgg's account along with other html attributes to make the img small and less noticeable.

This is one example:

Since CSRF countermeasures (__elgg_token, __elgg_ts) are switched off, the GET Request above will not need to include the right values and will work. I will create a sample index.html to be loaded by the malicious website http://www.csrflabattacker.com (Fig. 2.1) and embed the html img tag into the index.html as shown in Fig. 2.2. I saved this file to ***index-task2.html***. On the web page itself, Alice will not be able to see the image element because it is so small.

In Fig. 2.3, Alice has no friends before visiting the malicious website. With Alice logged in on one tab, we visit the malicious website http://www.csrflabattacker.com. In Fig. 2.4, we see the GET Request to add Boby as a friend is crafted and sent and in Fig. 2.5 we see that Alice now has added Boby as a friend.

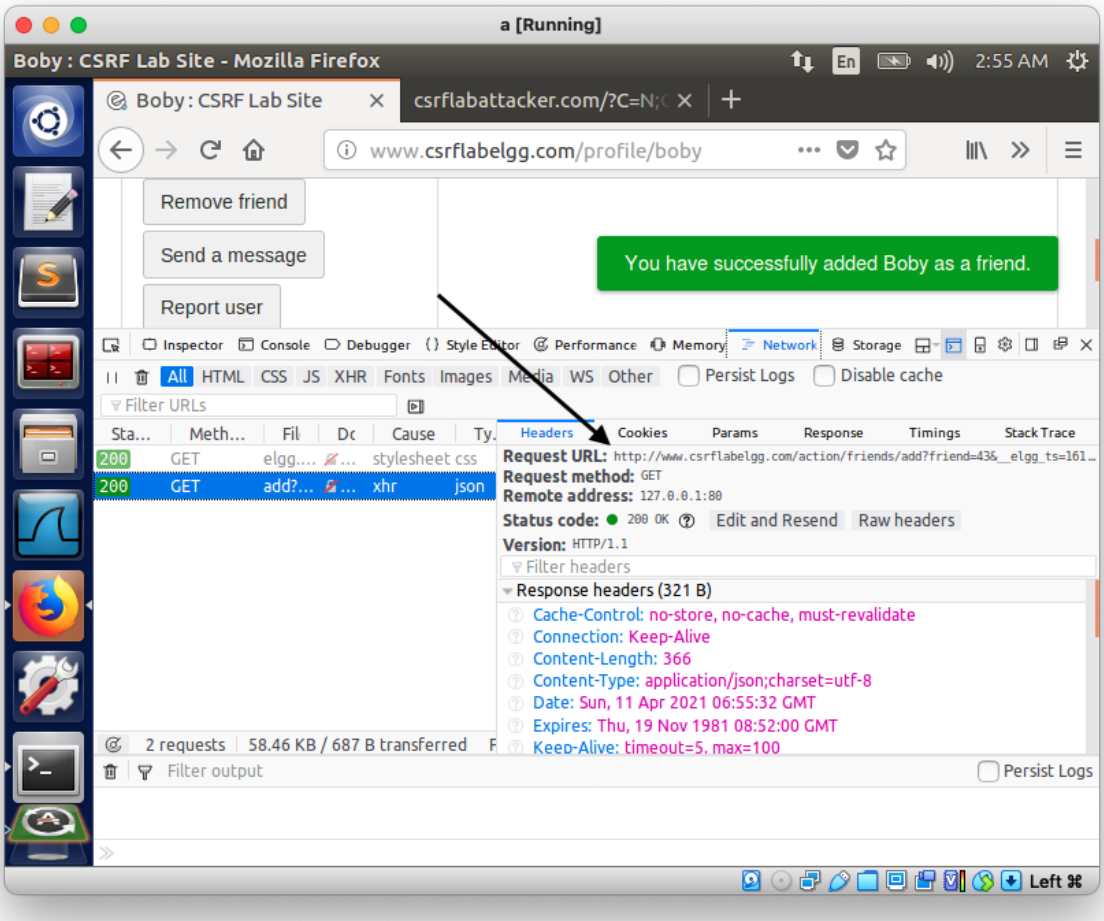

Fig 2.0 GET Request sent by Elgg's Add Friend Service

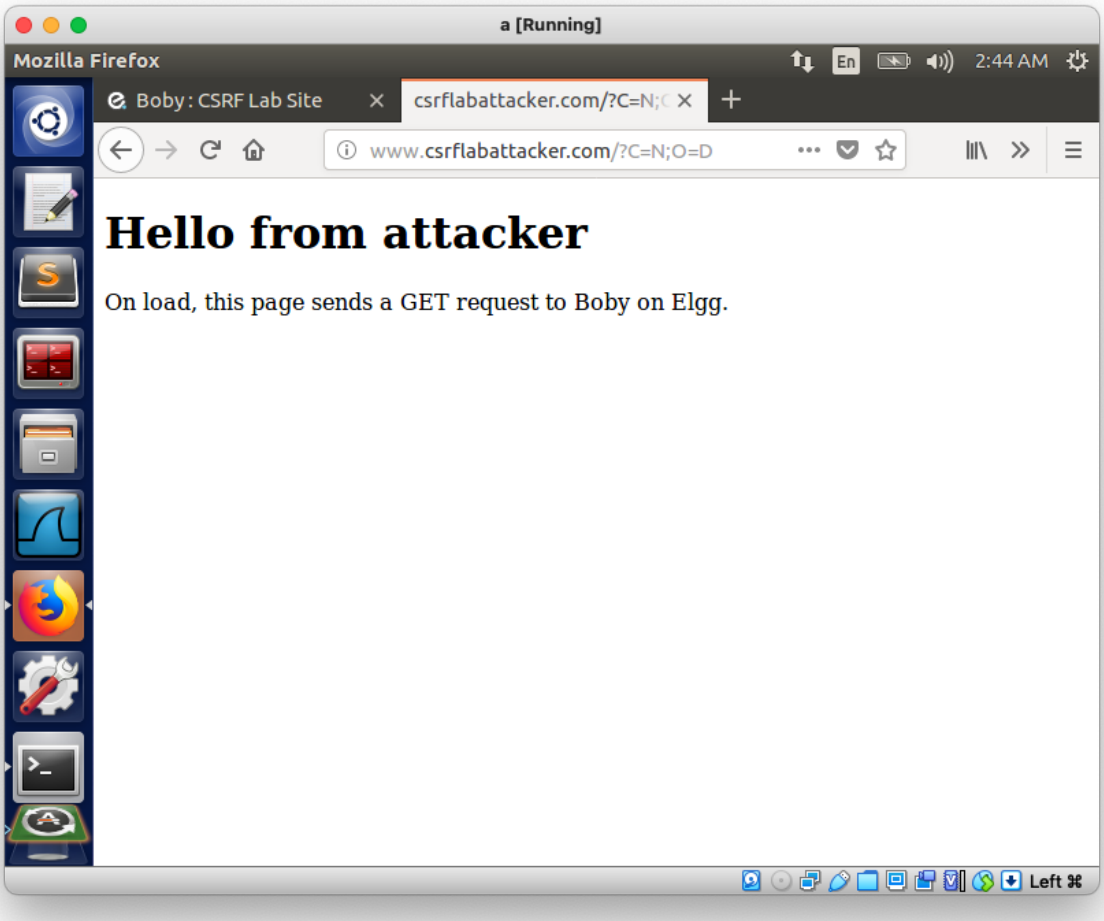

Fig 2.1 www.csrflabattacker.com with embedded html img tag

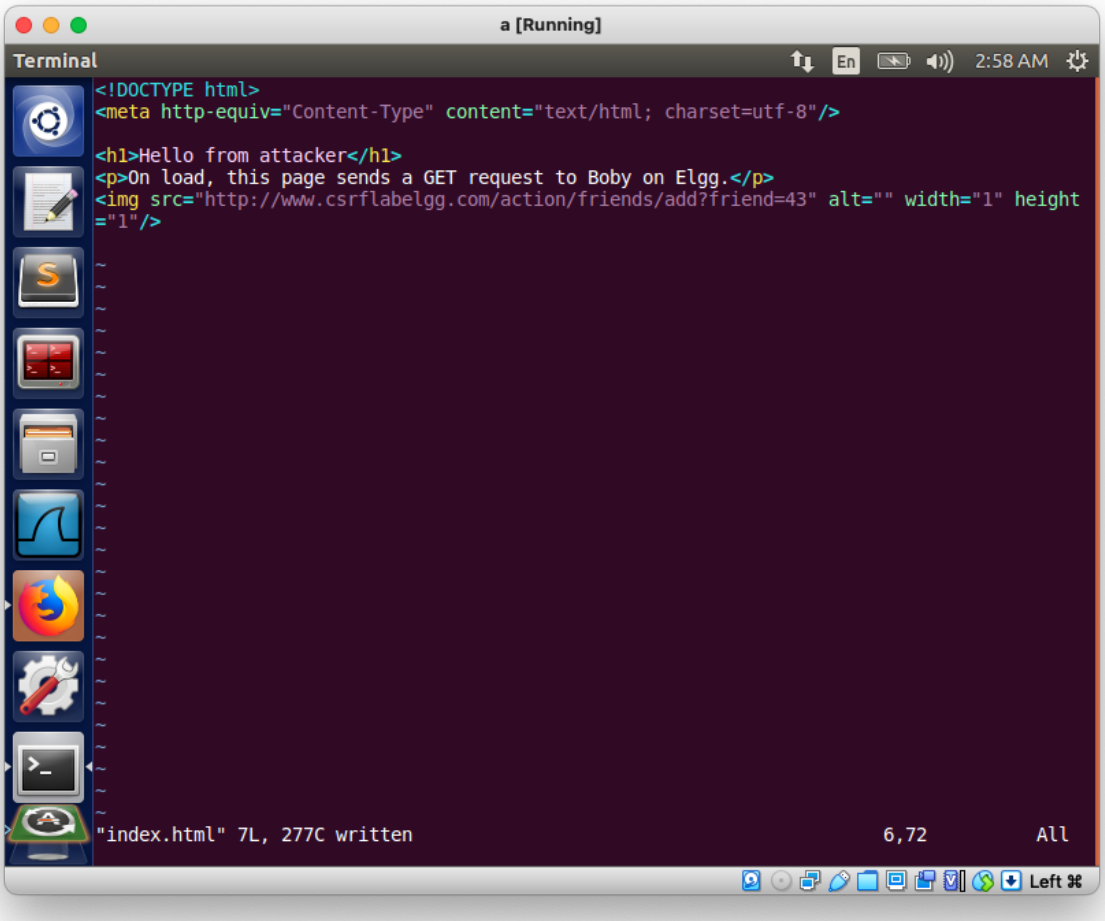

Fig 2.2 html code for http://www.csrflabattacker.com

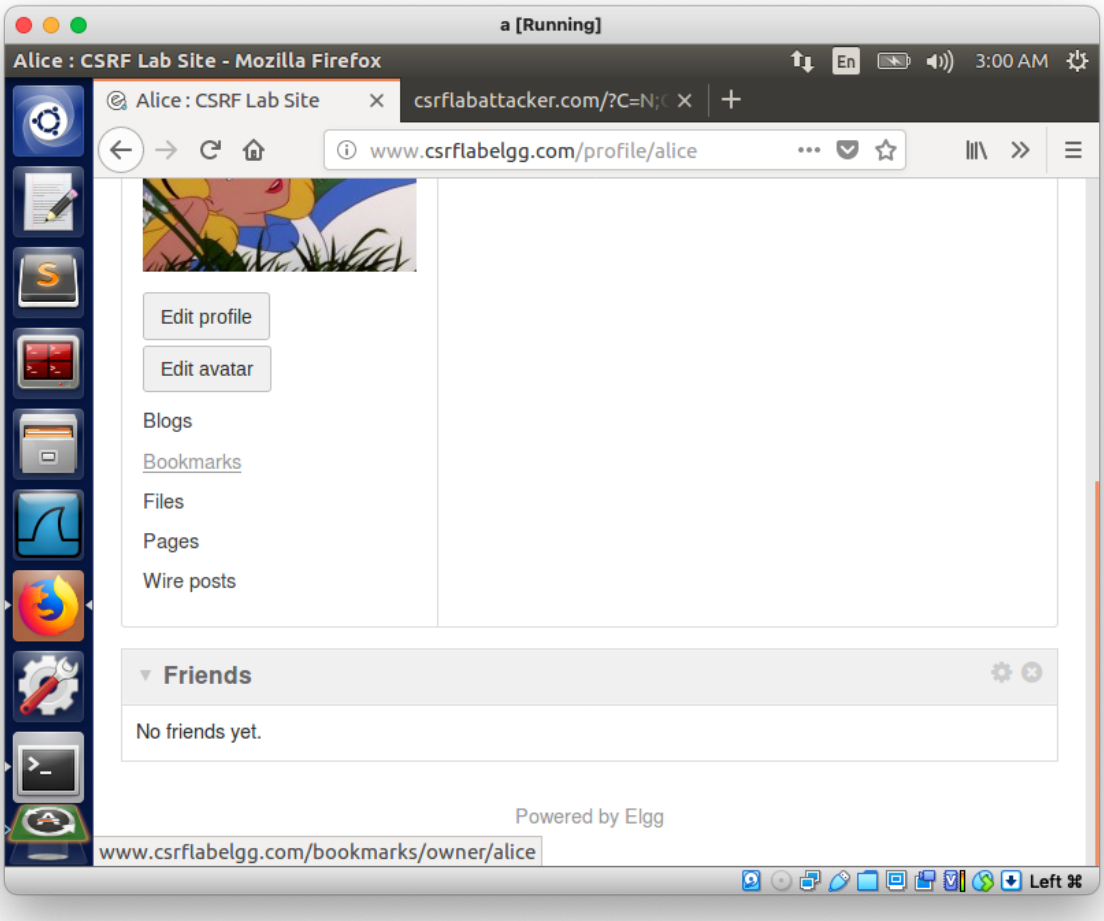

Fig. 2.3 Alice Friend List before visiting the malicious website

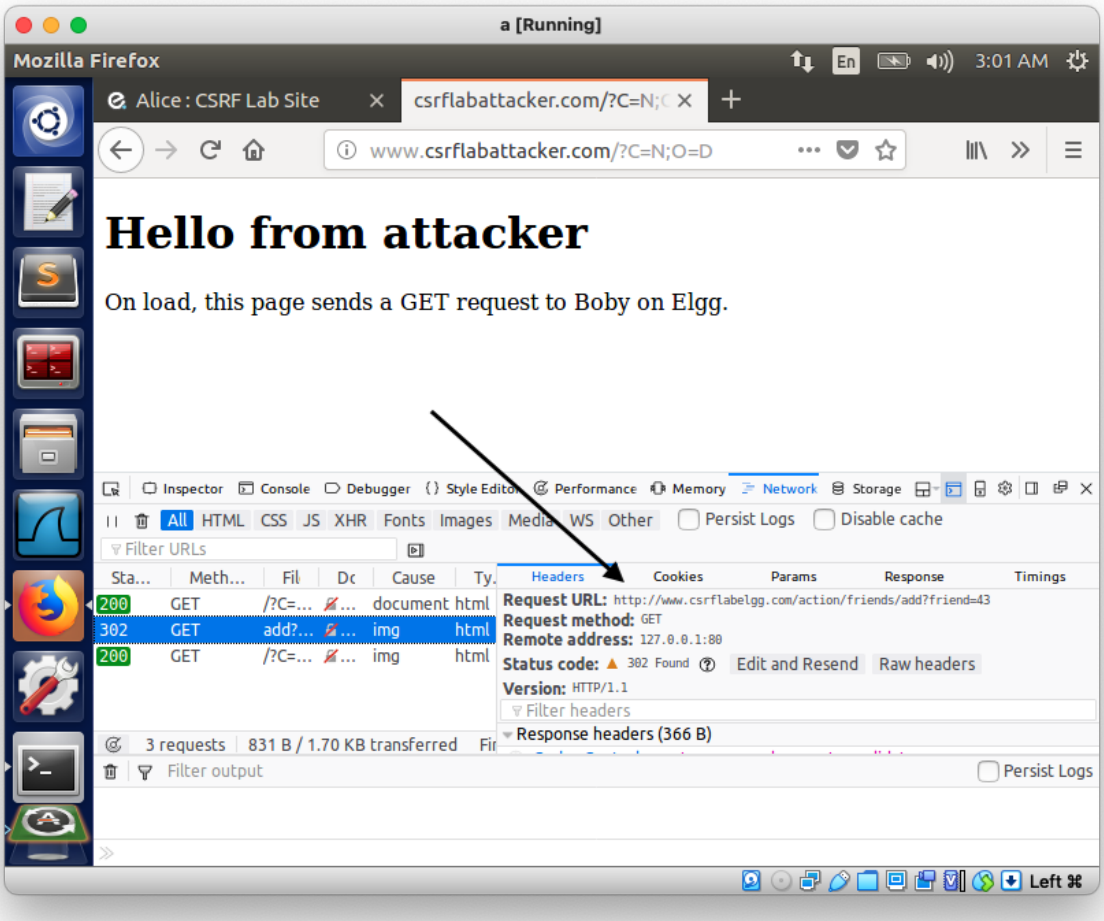

Fig 2.4 GET Request crafted to add Boby as a friend upon loading the webpage

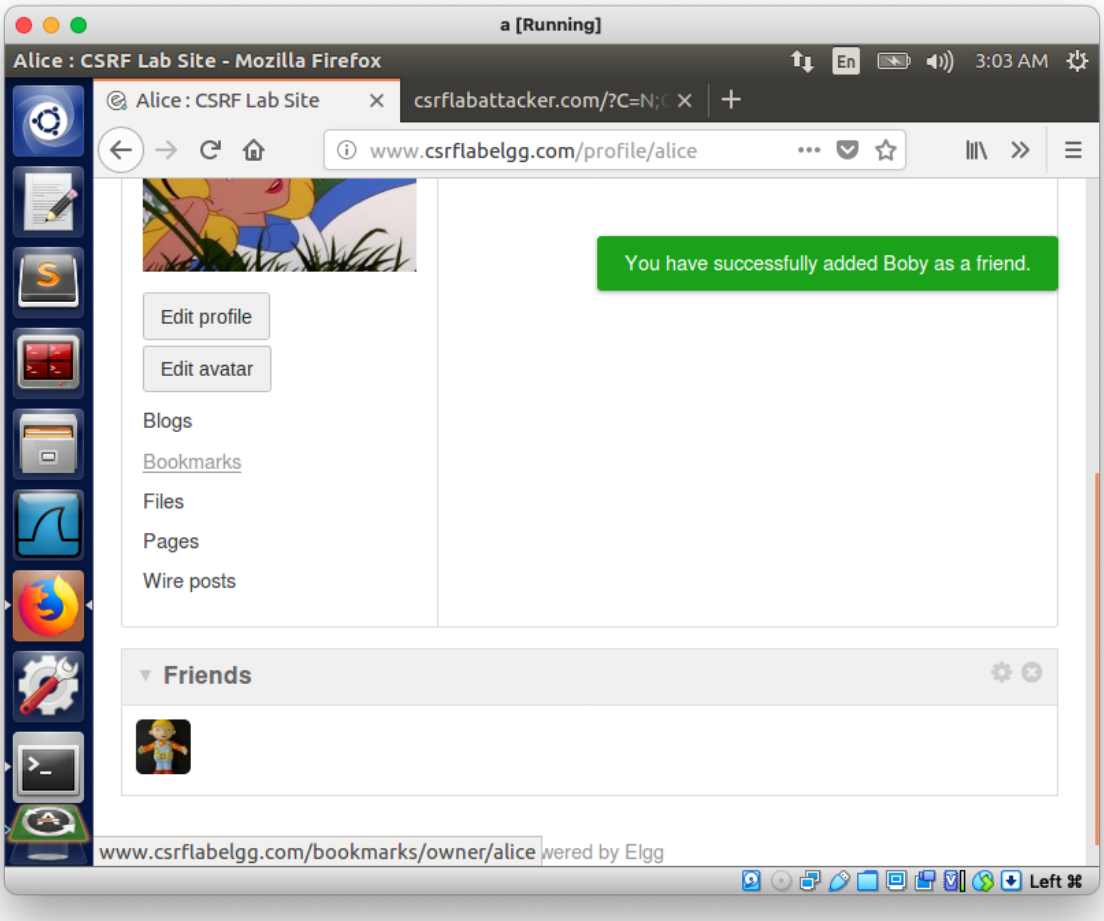

Fig. 2.5 Boby is now in Alice's friend list

Task 3: CSRF Attack using POST Request

The objective of your attack is to modify the victim's profile. In particular, the attacker needs to forge a request to modify the profile information of the victim user of Elgg.

In Fig. 3.0, we see the parameters used in Elgg's Edit Profile Service and we see the POST Request Headers associated with Elgg's Edit Profile Service in Fig. 3.1. Essentially, we need to craft a POST Request to http://www.csrflabelgg.com/action/profile/edit and attach the required parameters for a successful invocation of Elgg's Edit Profile Service. We can use the description field to save "Boby is my Hero" on the Victim's Profile.

Now, let us craft the malicious website that will perform the CSRF Attack via a POST Request. I will save the code to a file called ***index-task3.html*** as seen in Fig. 3.2. I will explain the code

logic as follows:

- The Javascript function ***forge_post()*** is automatically triggered when the page is loaded
- ***forge_post()*** creates a hidden form, with
    - the description entry filled with ***Boby is My Hero***
    - its accesslevel entry set as 2 (public)
    - guid entry filled as 42 (Alice's GUID)
    - Name entry filled as the victim "Alice"

When a victim (Alice) visits the page, the form will automatically be submitted from the victim's browser to the edit-profile service at http://www.csrflabelgg.com/action/profile/edit. This causes the message ***Boby is My Hero*** to be added to the victim's Profile Page.

With Alice logged in on one tab, we get Alice to click on the malicious website http://www.csrflabattacker.com and the Javascript function will load, crafts and sends a malicious POST Request (Fig. 3.3) and add to her Profile Page ***Boby is My Hero*** (Fig. 3.4).

Question 1: 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 use social engineering to find out if Alice leaked her Elgg's GUID online.

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.

Boby cannot launch this attack with anybody who visits his malicious web page since the GUID of each user is different and only when the GUID is specified correctly will the attack be successful.

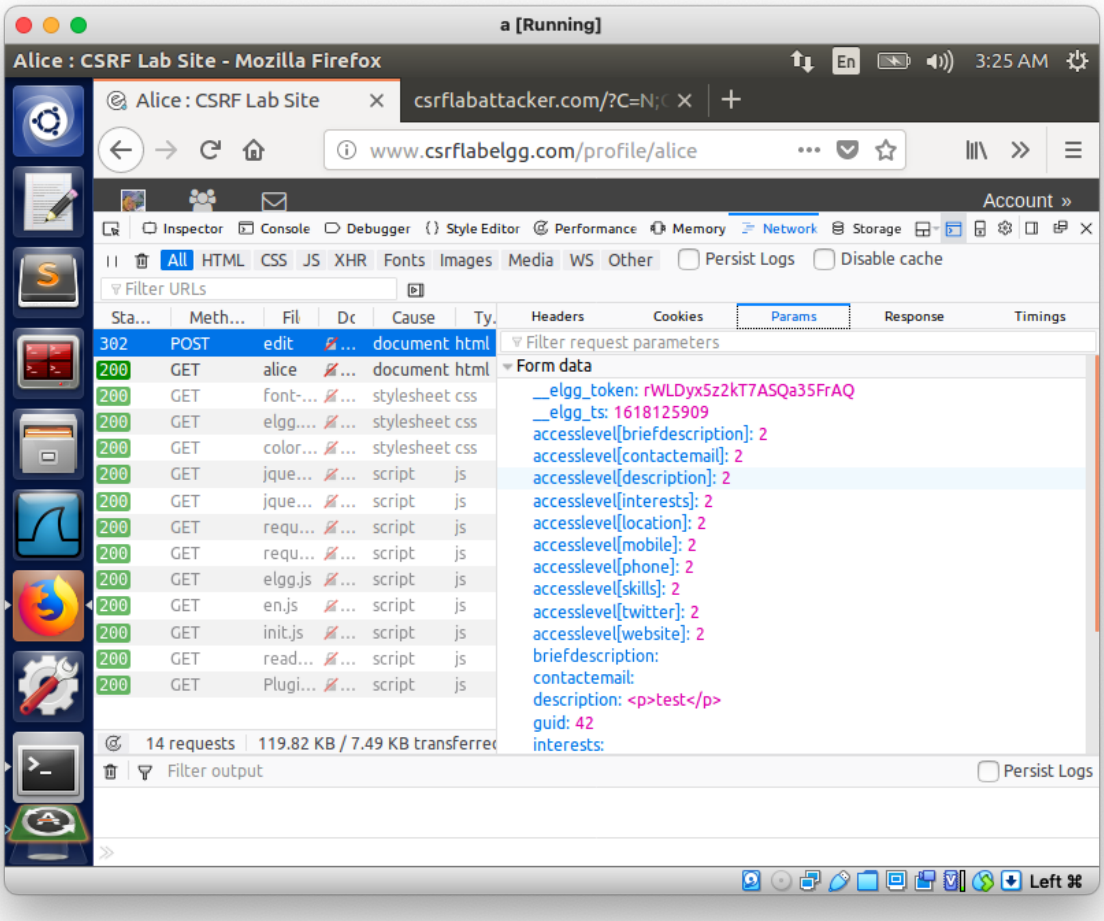

Fig. 3.0 Parameters associated with POST Request for Elgg's Edit Profile Service

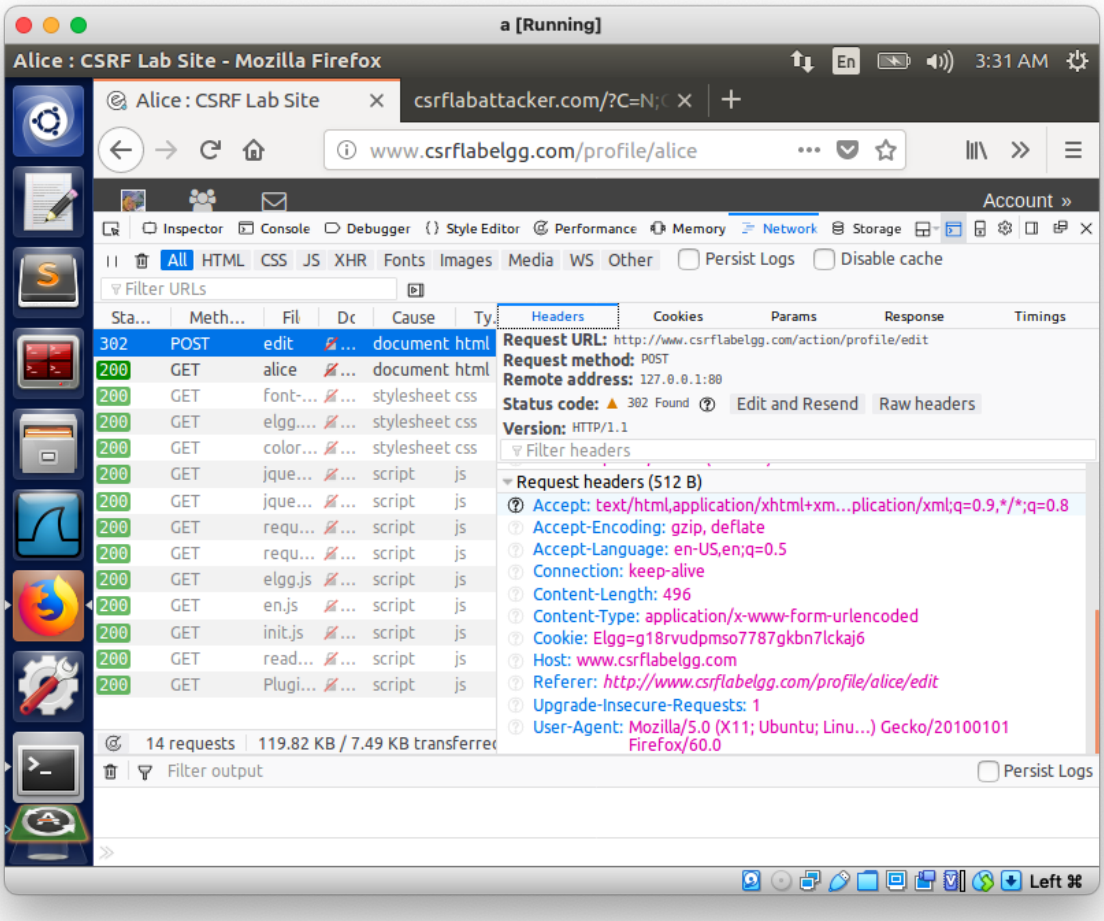

Fig. 3.1 POST Request Headers for Elgg's Edit Profile Service

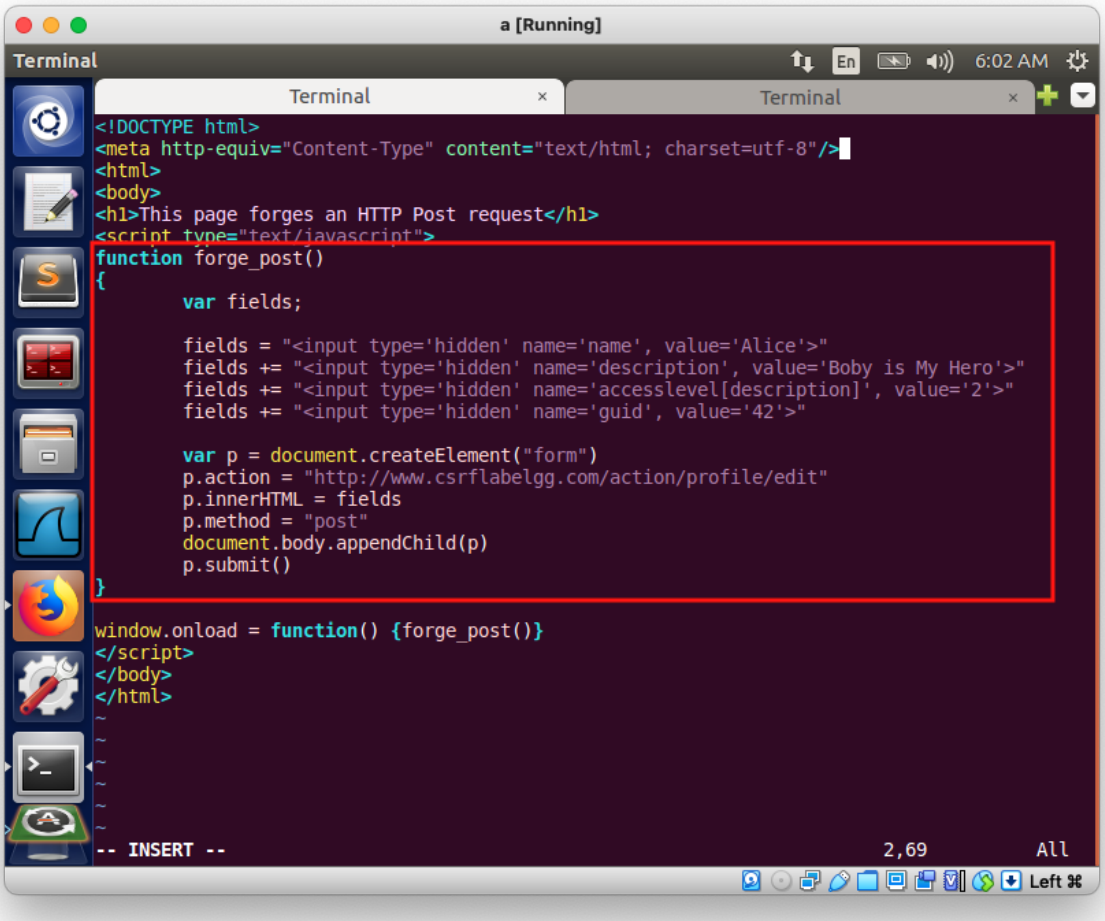

Fig. 3.2 index-task3.html

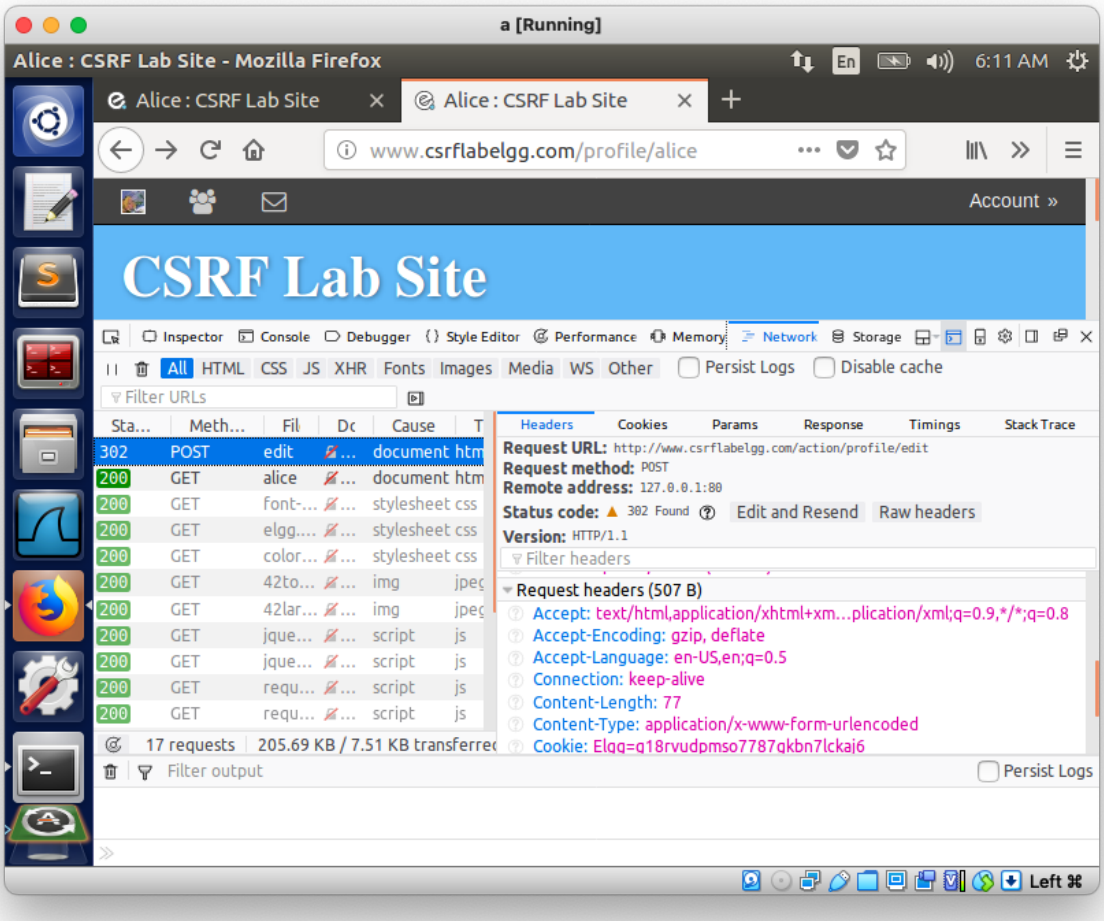

Fig. 3.3 Malicious POST Request crafted and sent

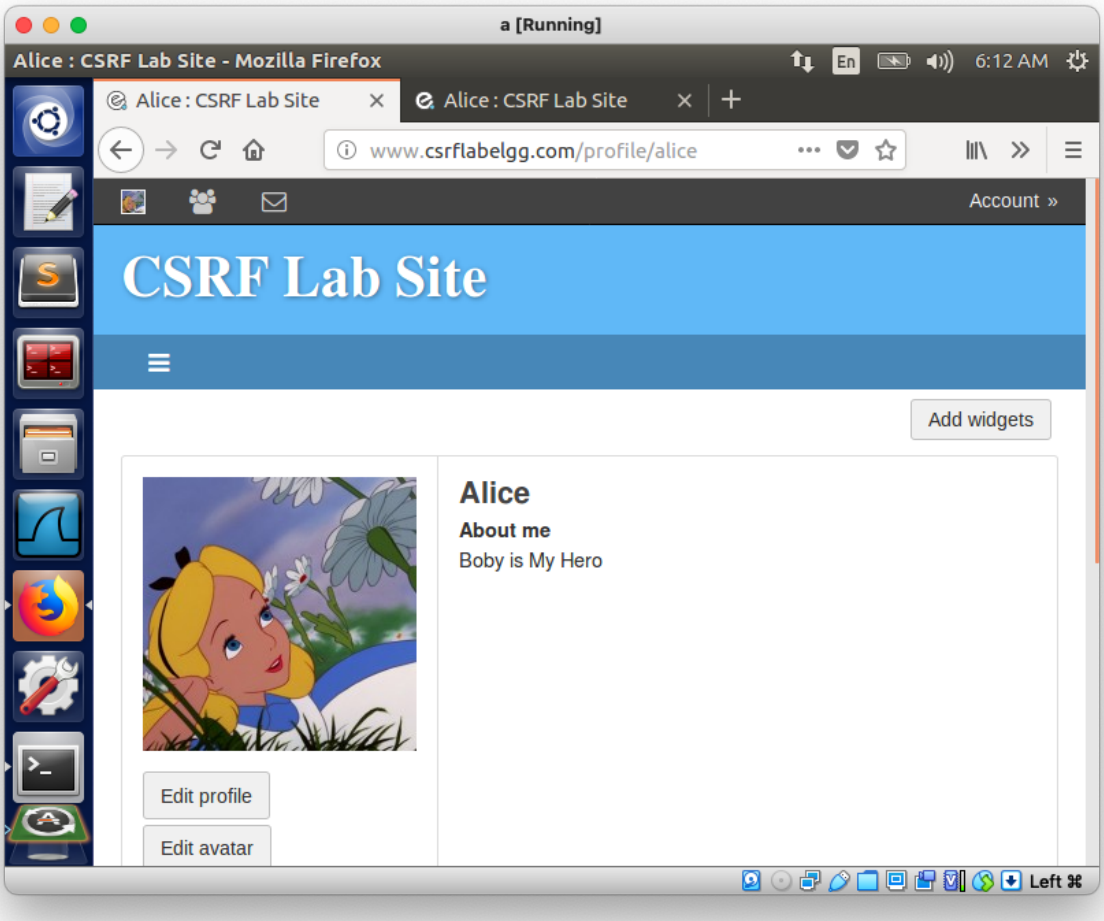

Fig. 3.4 Alice's Profile Page now has the string ***Boby is My Hero***

Task 4: Implementing a countermeasure for Elgg

https://github.com/roflcer/Cross-Site-Request-Forgery-Attack/blob/master/Lab%208.pdf
https://www.studocu.com/en-us/document/temple-university/computer-architecture/lecture-notes/lab-8-csrf-attack-seedlab/8384623/view

Task: After turning on the countermeasure (Fig. 4.0), I will re-attempt CSRF attack, and describe my observation.

Re-Attempt in Task 2
With Alice logged into her Elgg account in one browser tab, we visit the malicious website http://www.csrflabattacker.com. Within Firefox Web Developer Tool's Network Tab, the GET Request is

successfully sent as seen in Fig. 4.3. As expected, we see alerts when reloading the Alice's browser tab that is logged into Elgg. The message in the alert is "**Form is missing __token or __ts fields"**. When we load Alice Profile Page, we do not see Boby under Alice's Friend list which is expected. This is seen in Fig. 4.4.

Re-Attempt in Task 3

With Alice logged into her Elgg account in one browser tab, we visit the malicious website http://www.csrflabattacker.com. Immediately, we see that the website keeps reloading and continuously resends the crafted POST Request. We see this behavior in a wireshark capture on the loopback interface in Fig. 4.1. Once Alice reloads the page that is logged into Elgg, we see notifications (red popups in Fig. 4.2). The message of the alert is "**Form is missing __token or __ts fields"** which is expected because the POST Request crafted by the malicious website did not include those fields. These fields are Elgg's countermeasures to CSRF.

For both tasks, The attacker is not able to get and send these secret tokens because the browser's access control prevents Javascript code in the attacker's page from accessing any content in Elgg's pages.

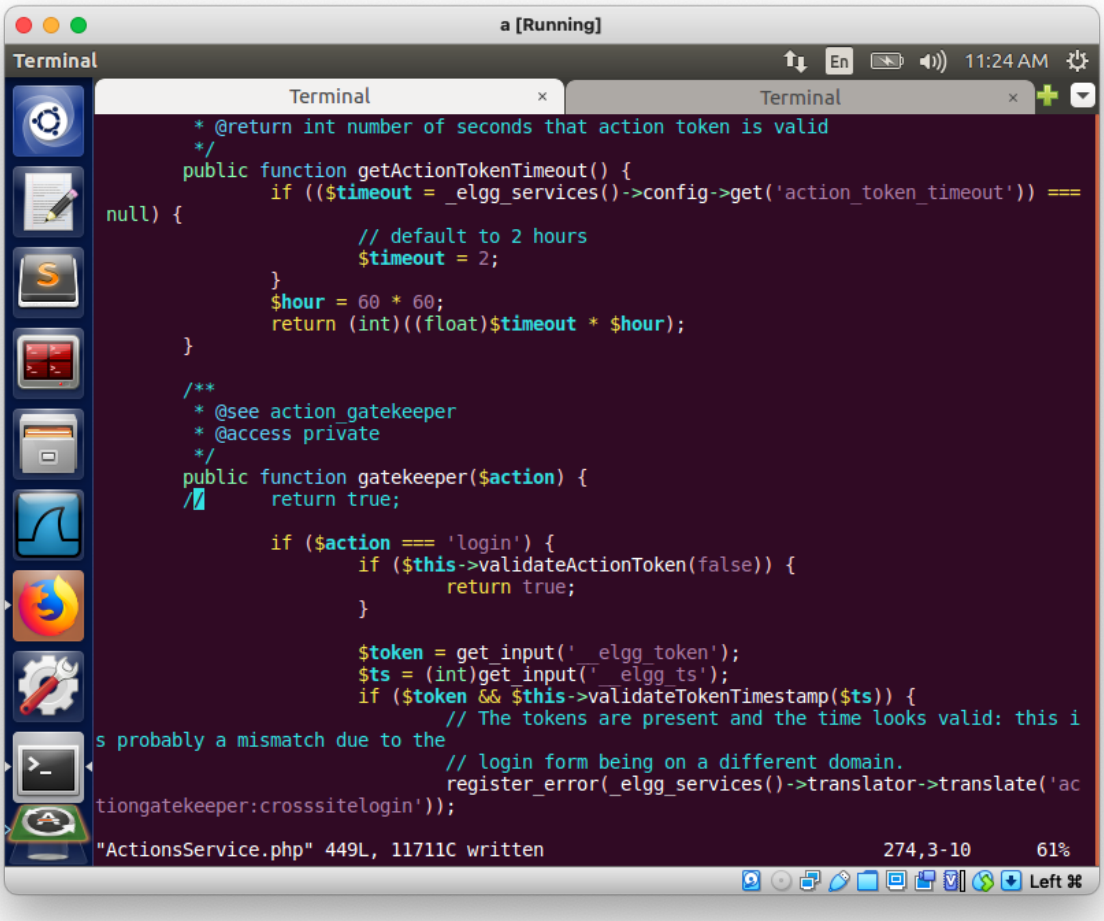

Fig. 4.0 Turning on Elgg CSRF countermeasure

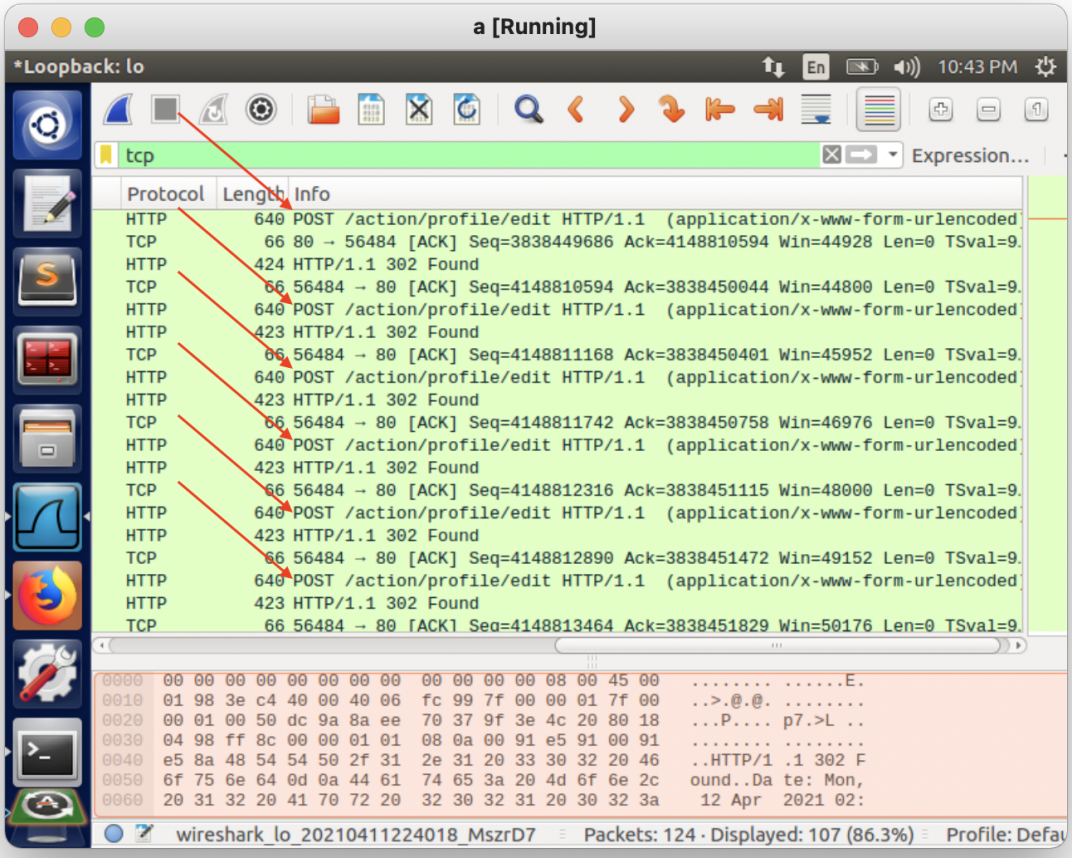

Fig 4.1 Repeated POST Requests sent by malicious website

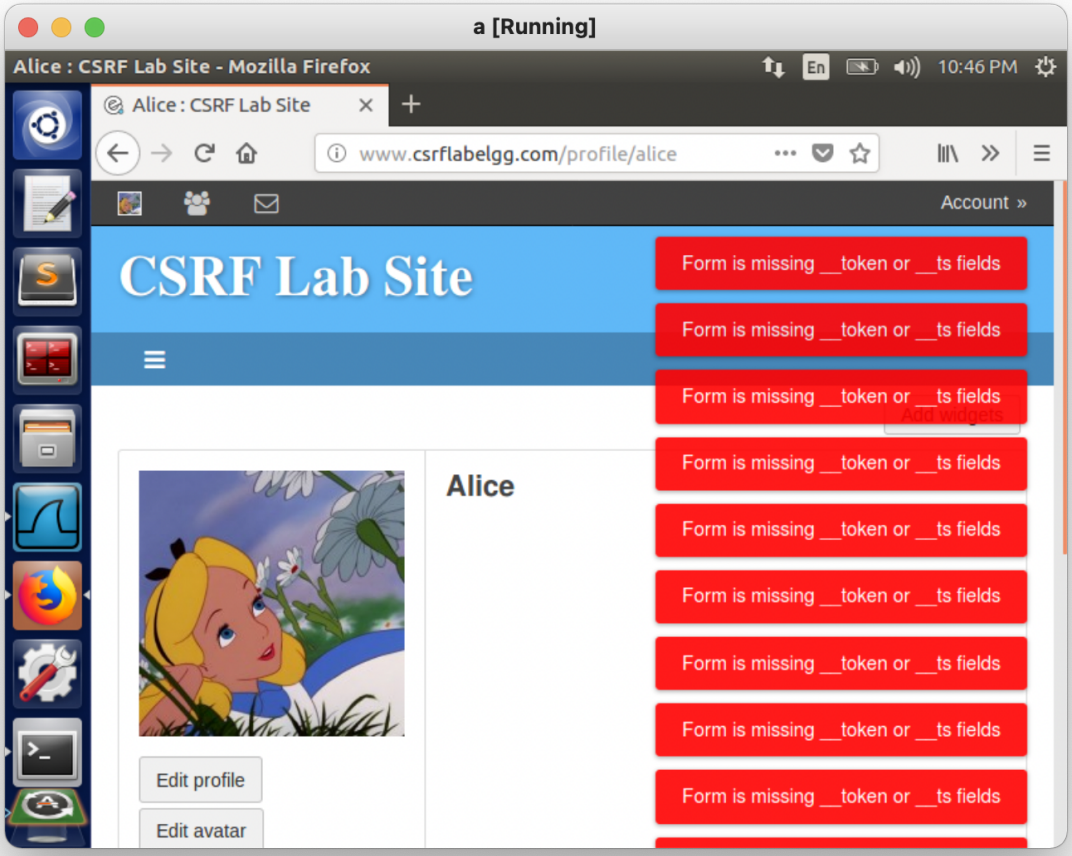

Fig. 4.2 Alert caused by the repeated POST Requests

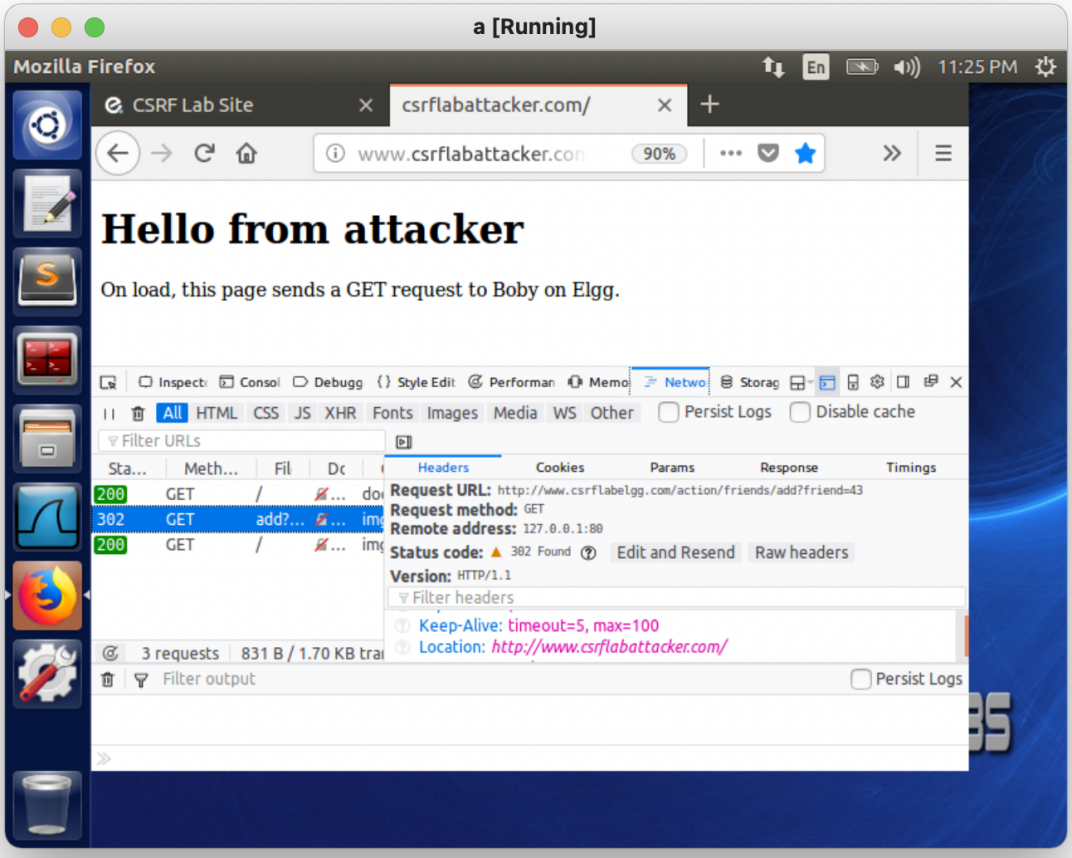

Fig 4.3 Successful GET Request sent to Elgg to add Boby as a friend

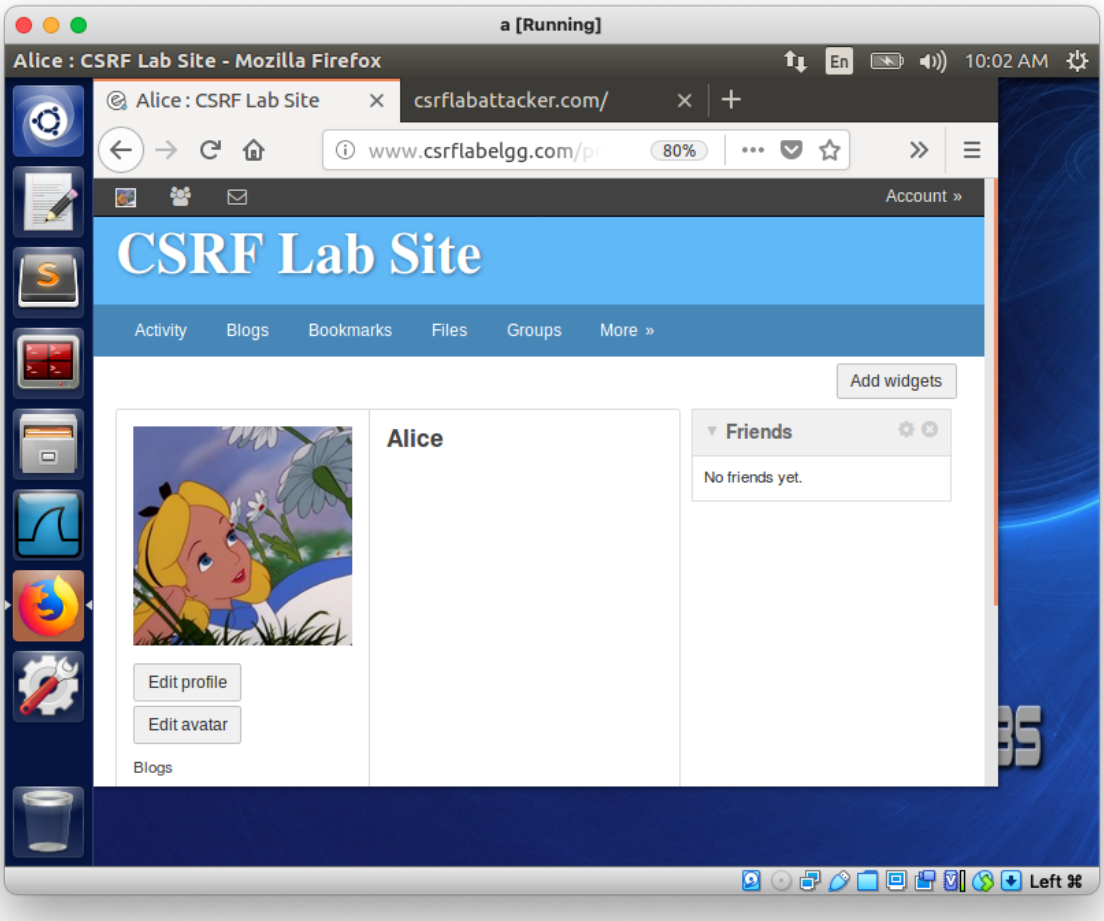

Fig. 4.4 Alice's Profile Page - Boby did not manage to add himself to Alice's friend list