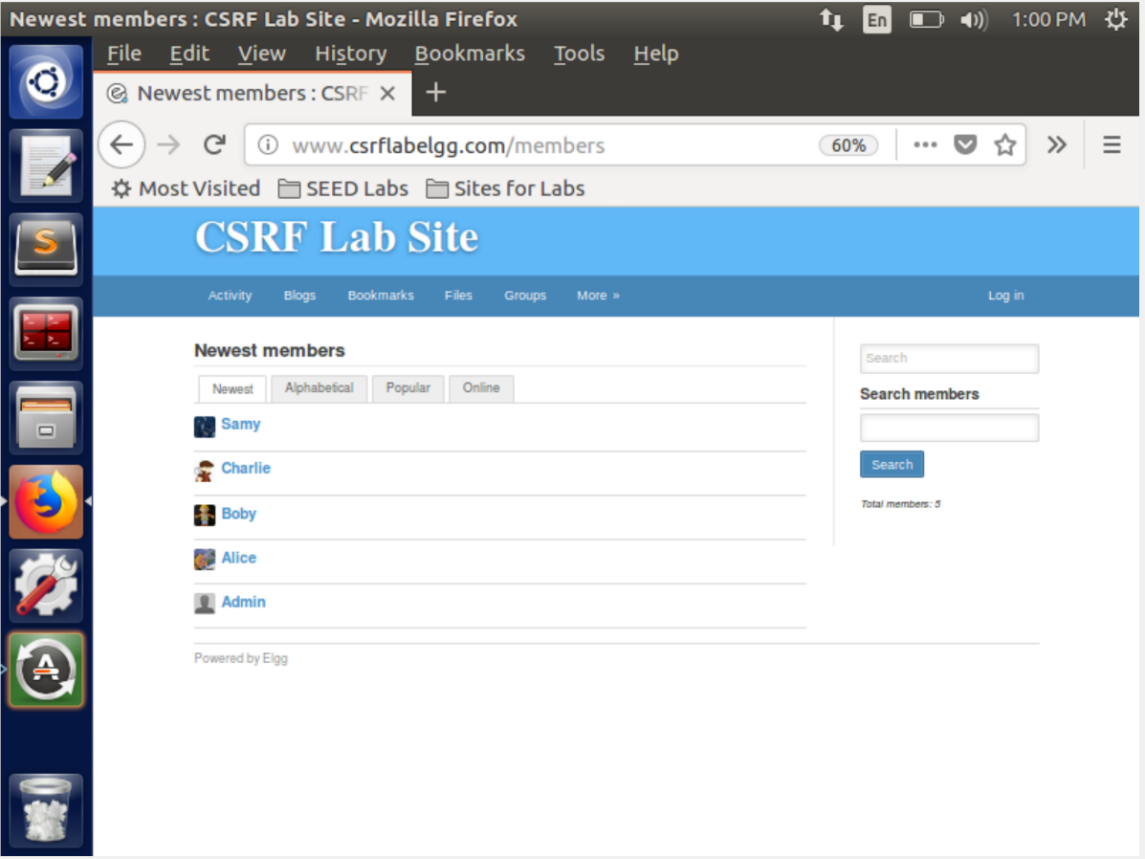
R.Rohith Rajagopalan

1001518031

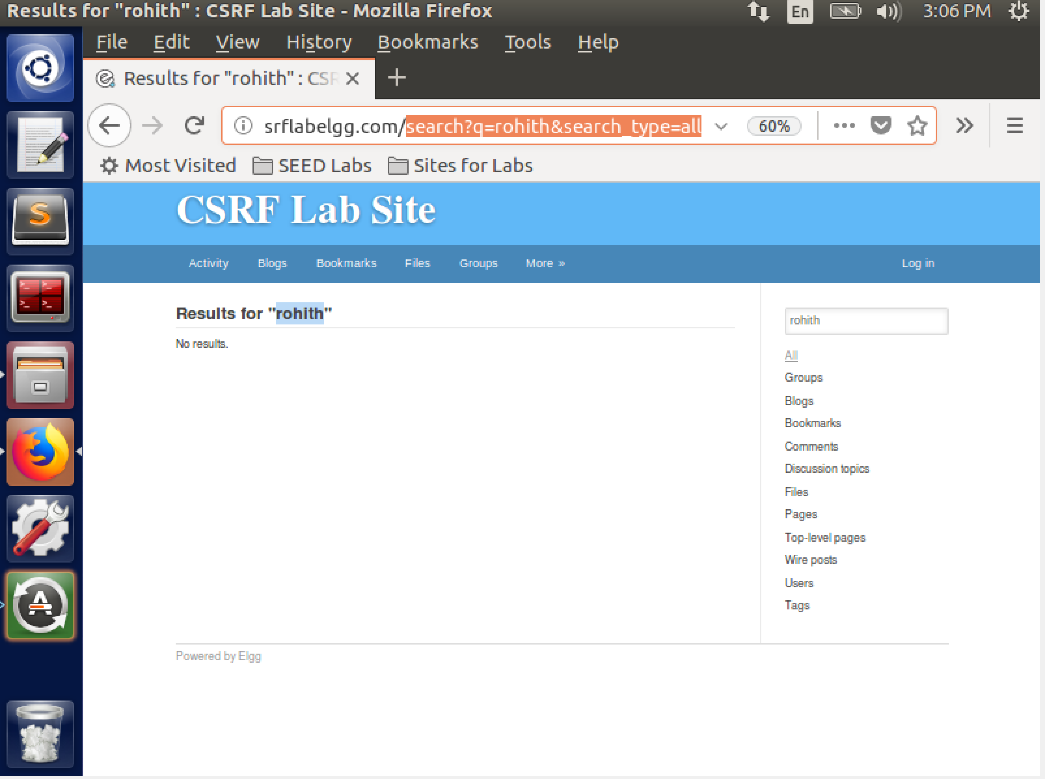
Cross-Site Request Forgery (CSRF) Attack

We perform the lab exercises on the following website given in the lab manual



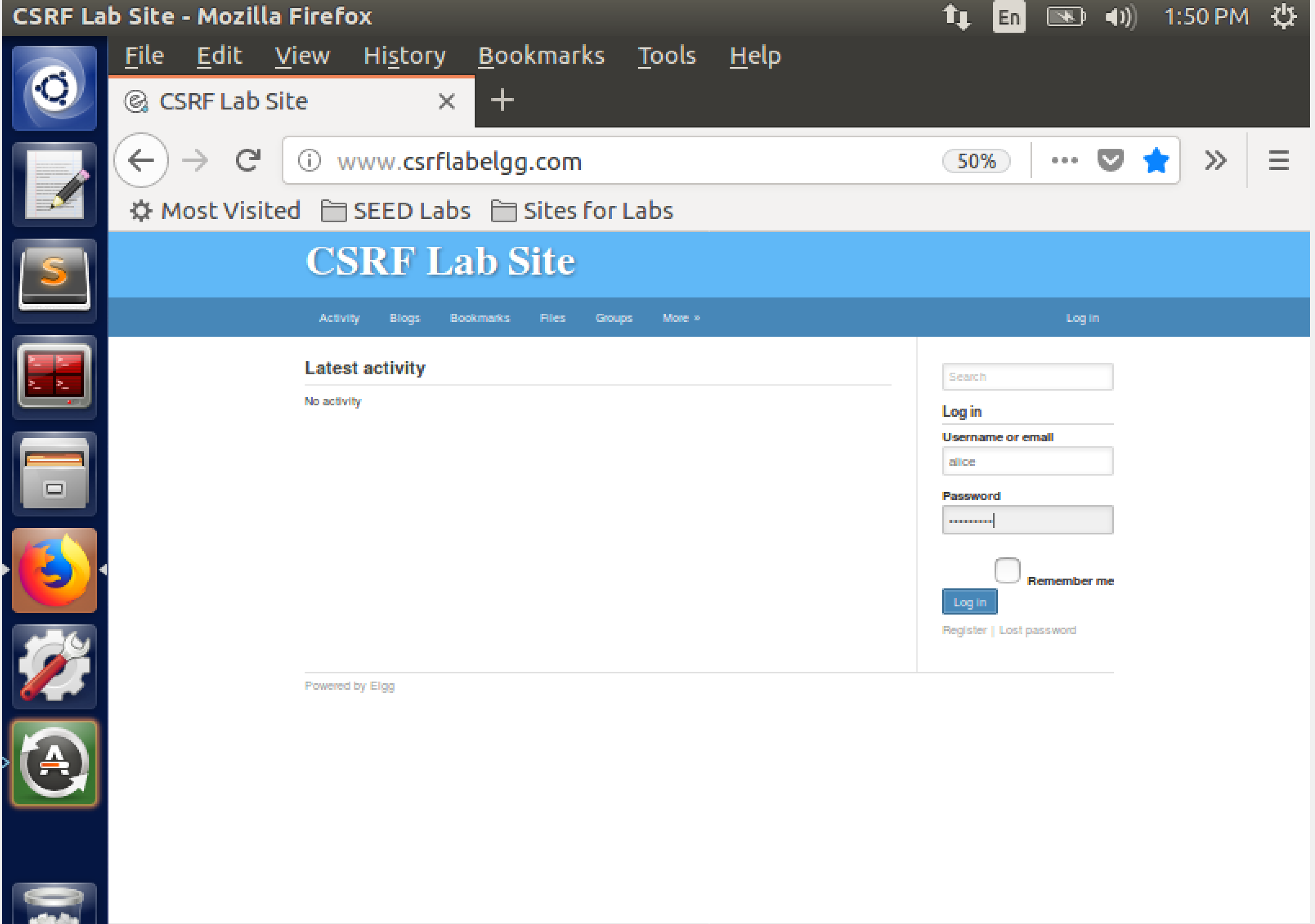
**Task 1: Observing HTTP Request.**

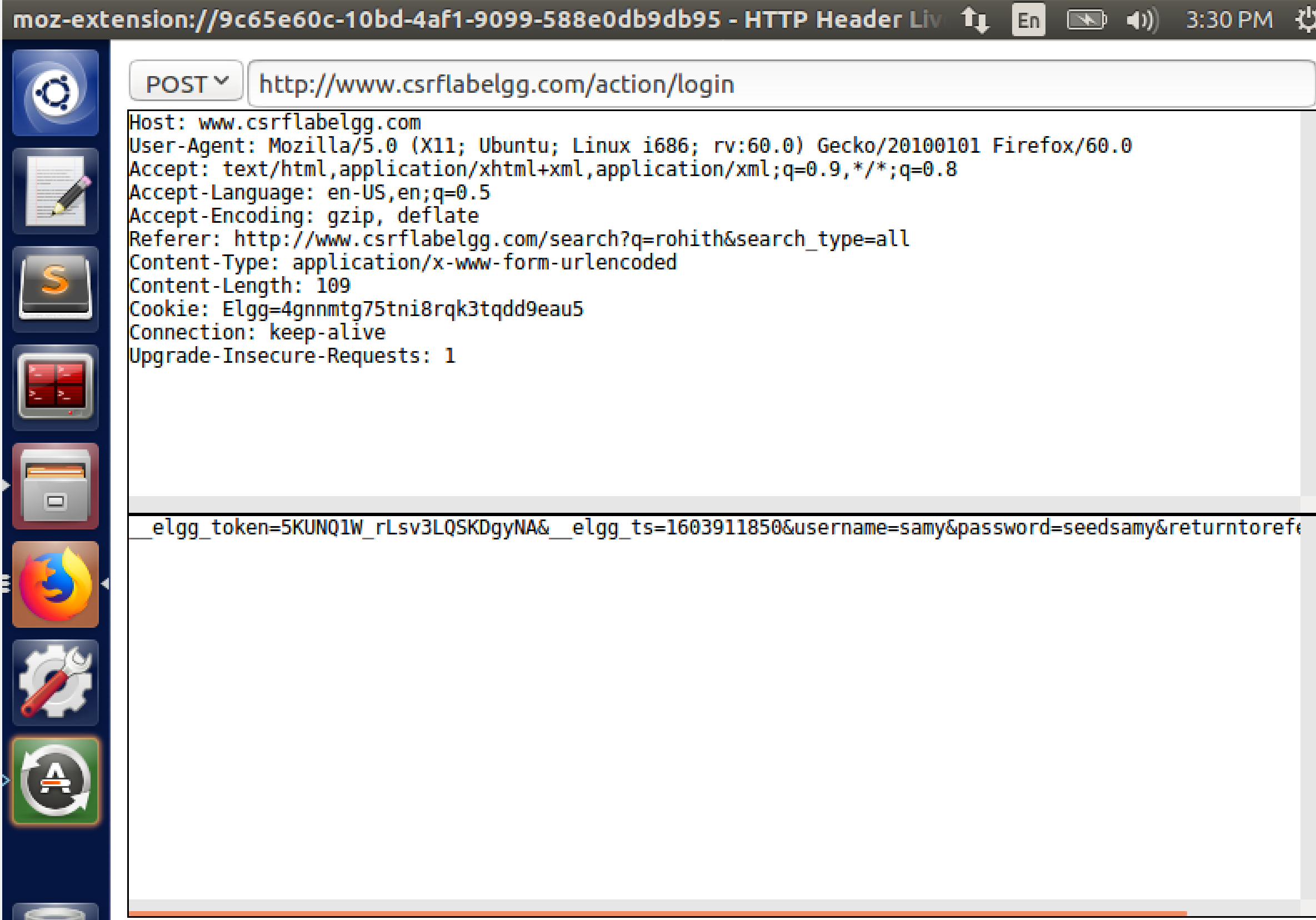
Here we observe the HTTP request. Here I search for my own name and see the HTTP request generated.



We observe that the method for the HTTP request is GET. We also see that the params sent in this request, the value of q is the string searched by us. We see that the accept fields are also sent to tell the server the type of data accepted. We see the param sent in the link as shown.

Next we perform another activity to observe the behavior of the POST request.



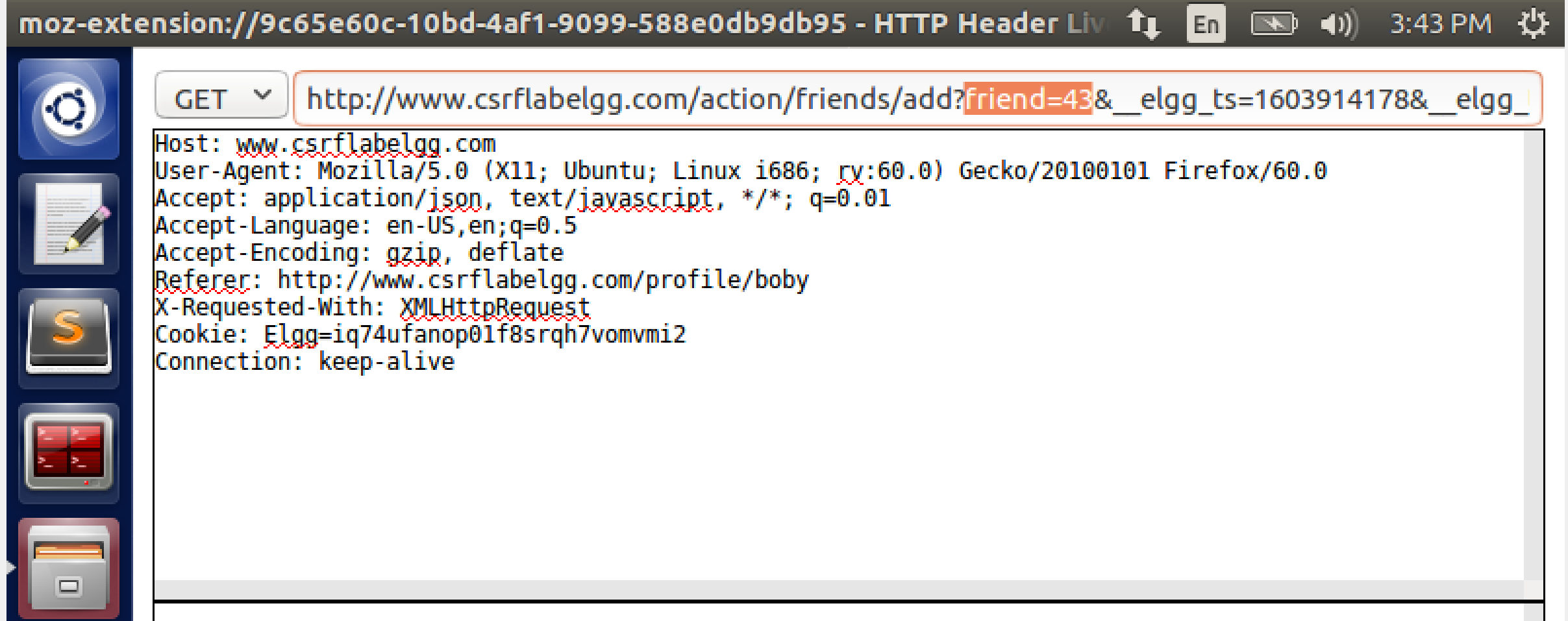


Here we see a post request along with the header information. There are details like content length and content type which were not in get method present in the post method. The username and password are the data entered by us are seen in the \_elgg\_token.

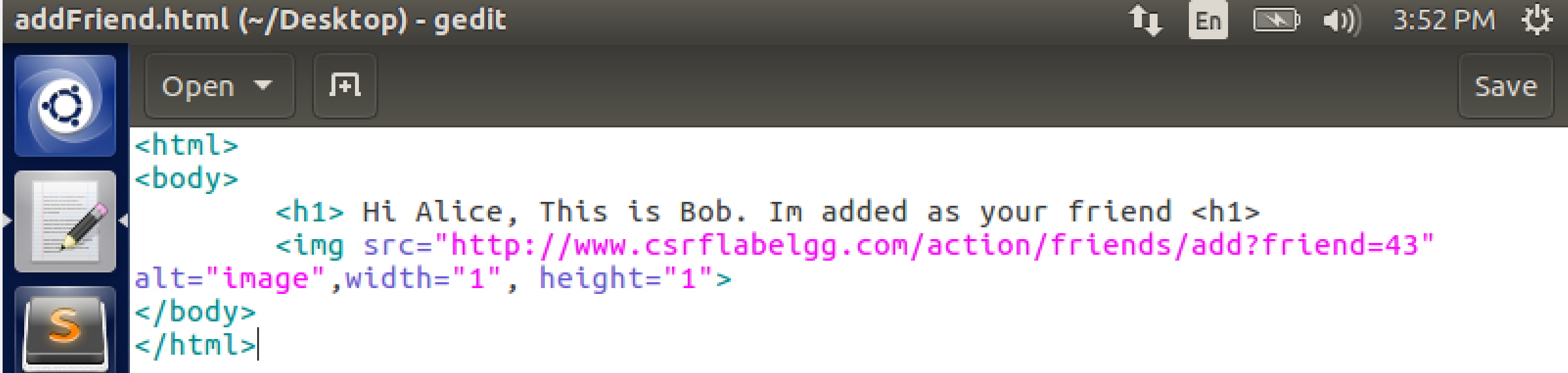
The key difference between both the methods is the Get method has all the params in the URL string and the POST method includes it in the body of the request this it has the content type and length in the header.

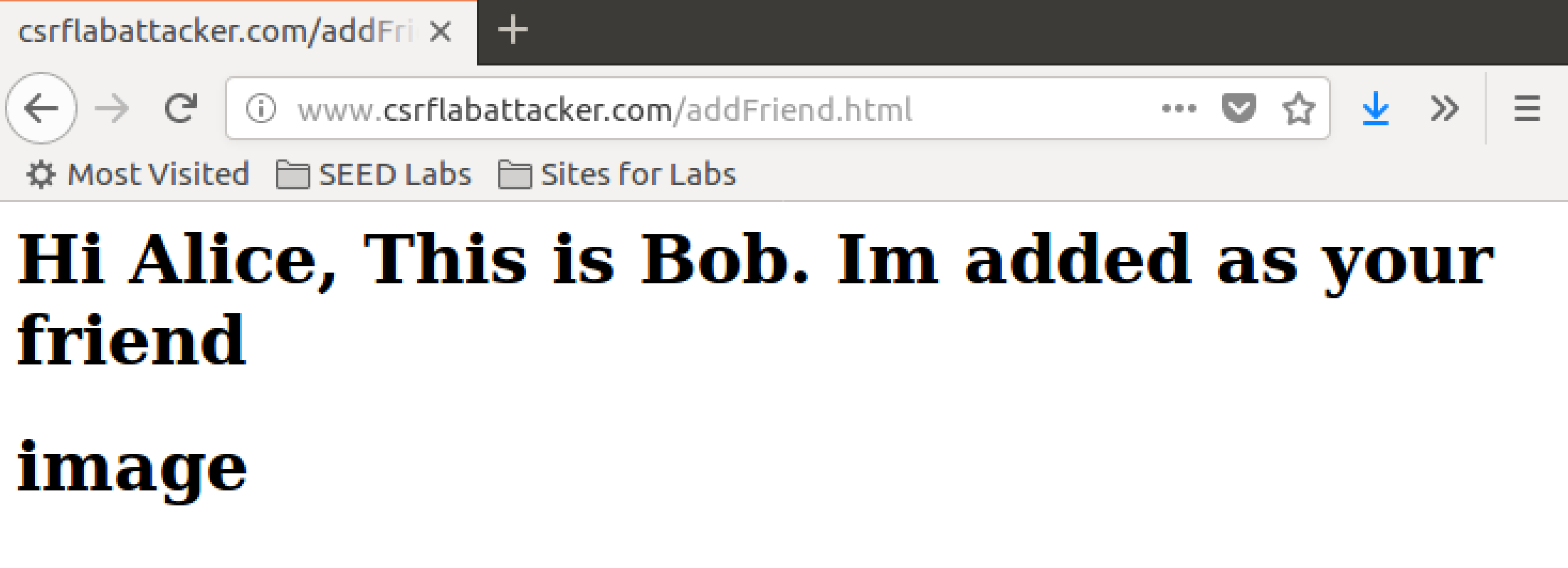
**Task 2: CSRF Attack using GET Request**

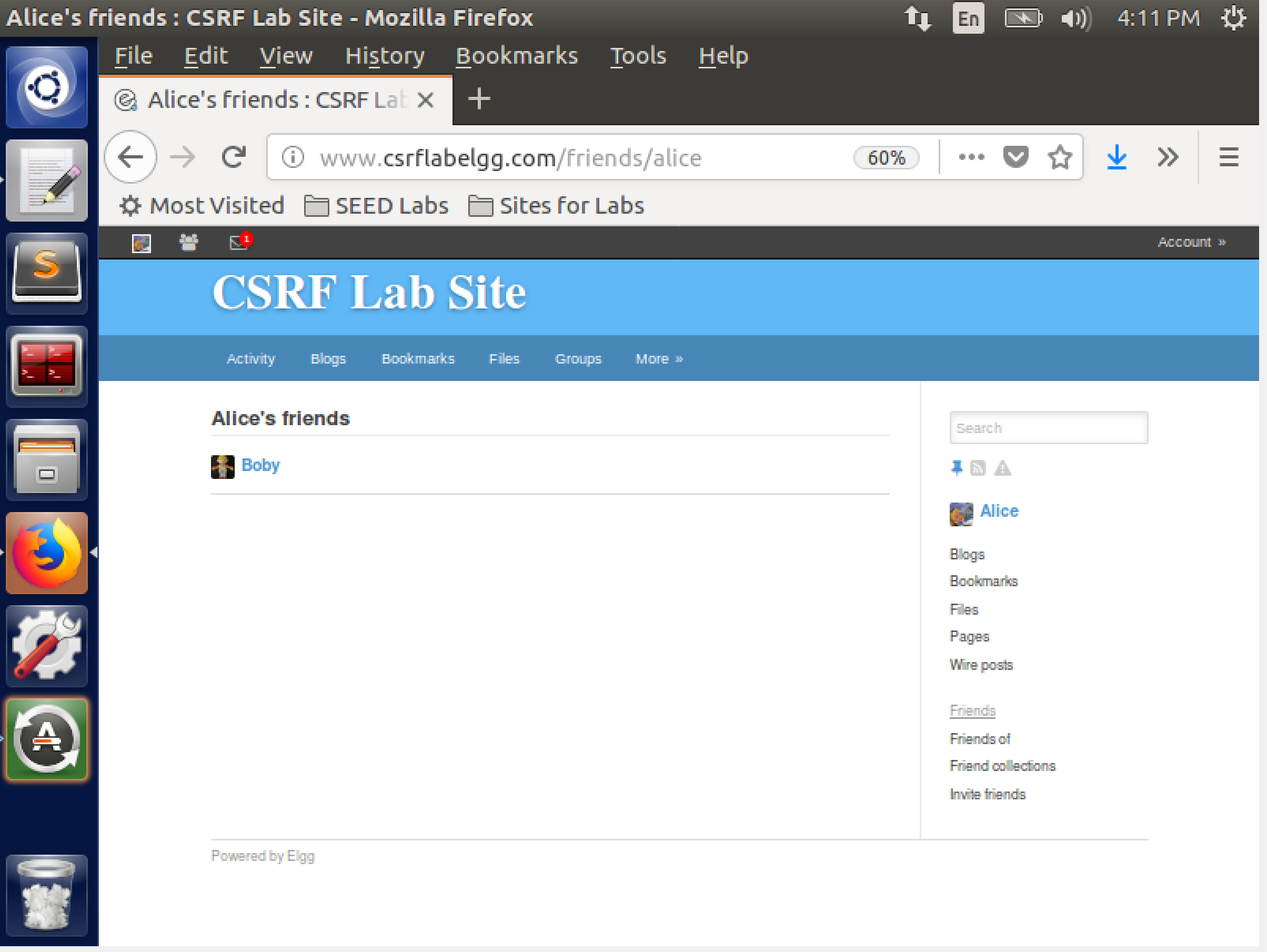
We log into charlie's account and add bob as a friend to observe the request parameters. From the params tab we see that friend had a value of 43 .



The GUID of the page owner is 43 and the page owner in our case is boby thus we can confirm that bobby is the page owner. We then try to create a web page that will add boby as Alice's friend. We will use the same request parameters similar to adding boby to Charlie but with some changes in the cookies and tokens. We use the image tag of HTML pages which sends an GET request as soon as the image is displayed. Since the countermeasures elgg\_ts and elgg\_token are disabled adding them here will have no effect. The file is created in the var/www/CSRF/Attacker with name addFriend.html, since this directory is linked with the attack page.We log in as alice as one of the requirements for the attack is that the victim should have an active session. First we see that she does not have any friends added. We then visit the website sent by bob to alice.





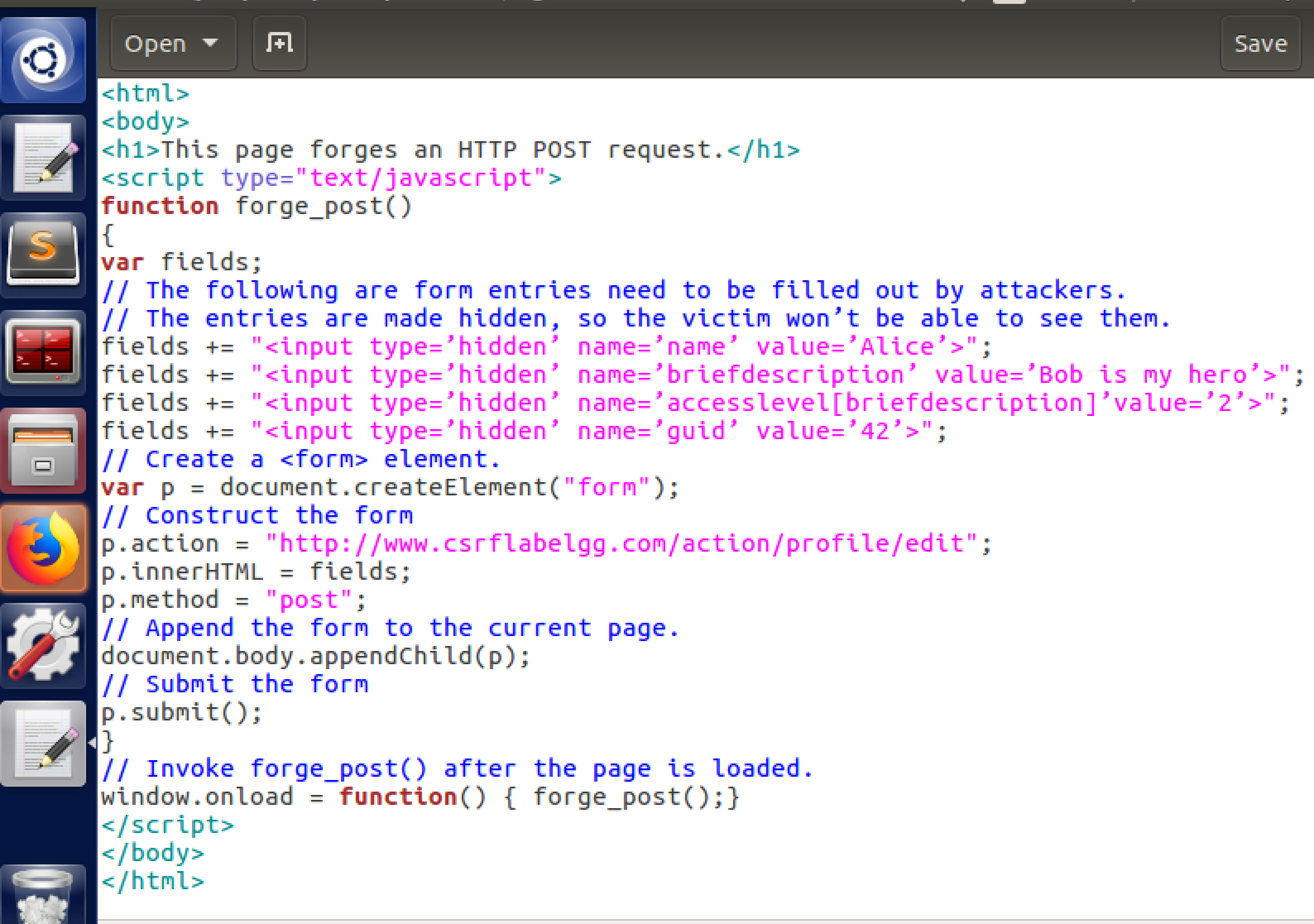


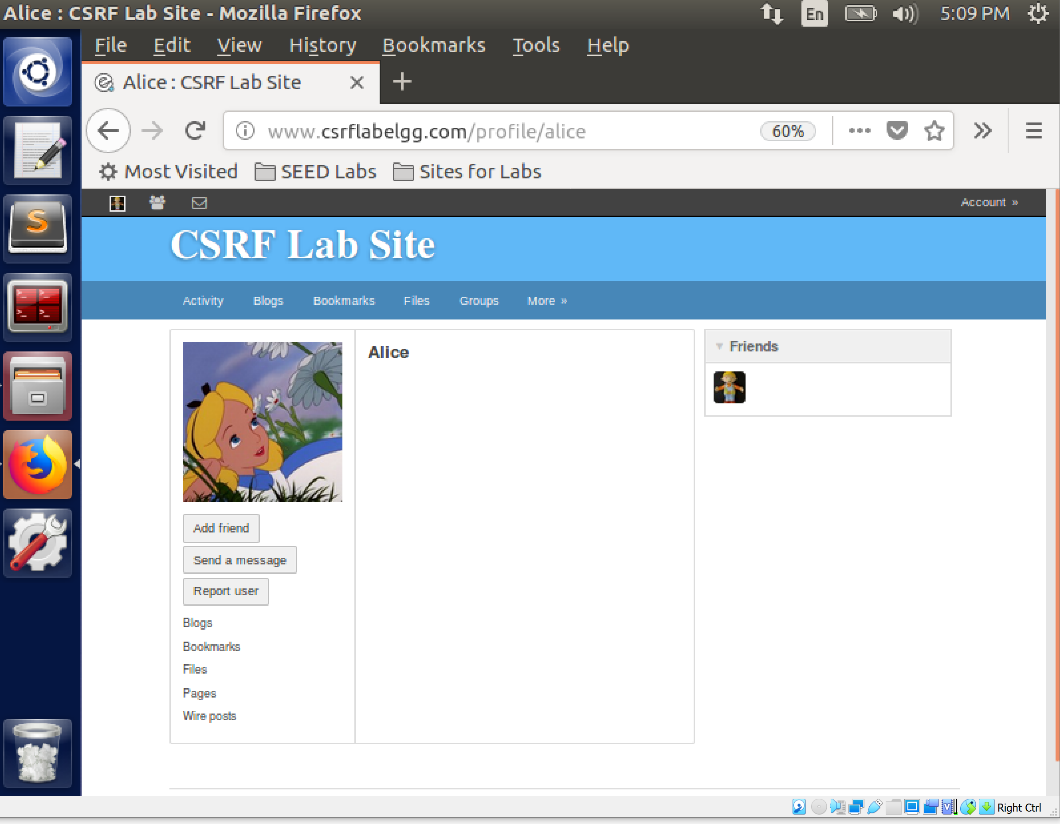


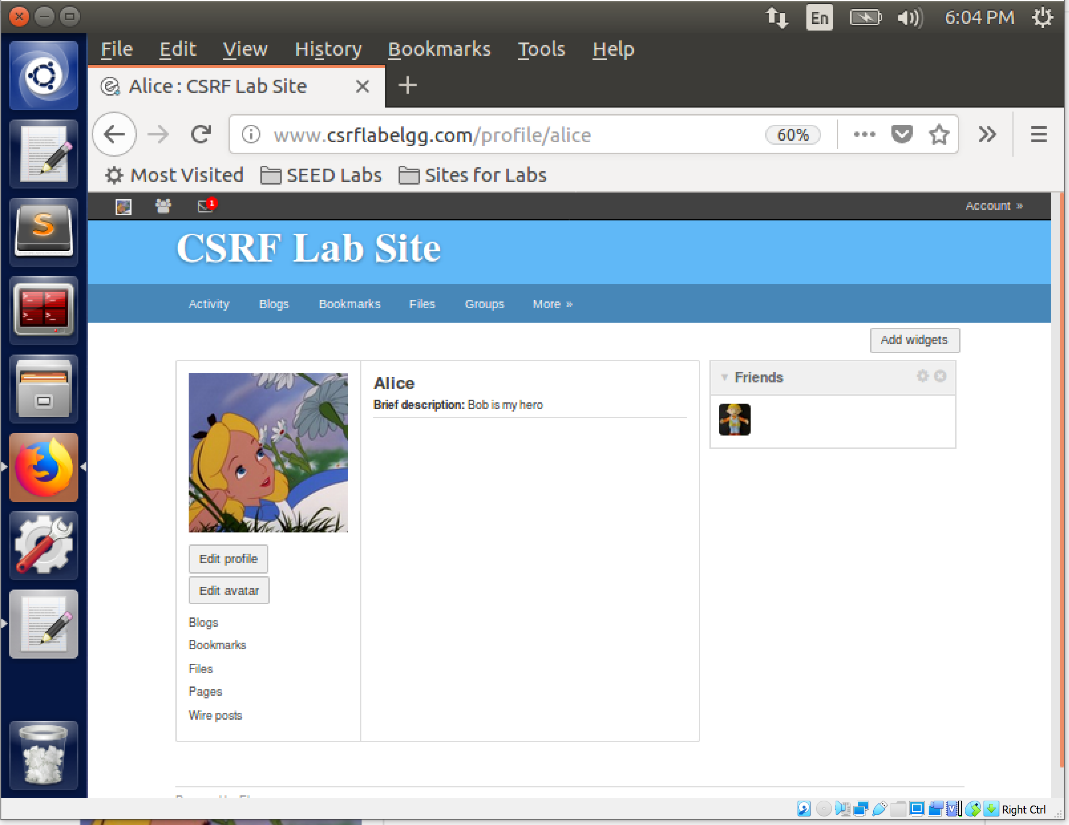
Thus we see that our attack is successful in adding bob as alicies friend without any intention of alice to add bob.

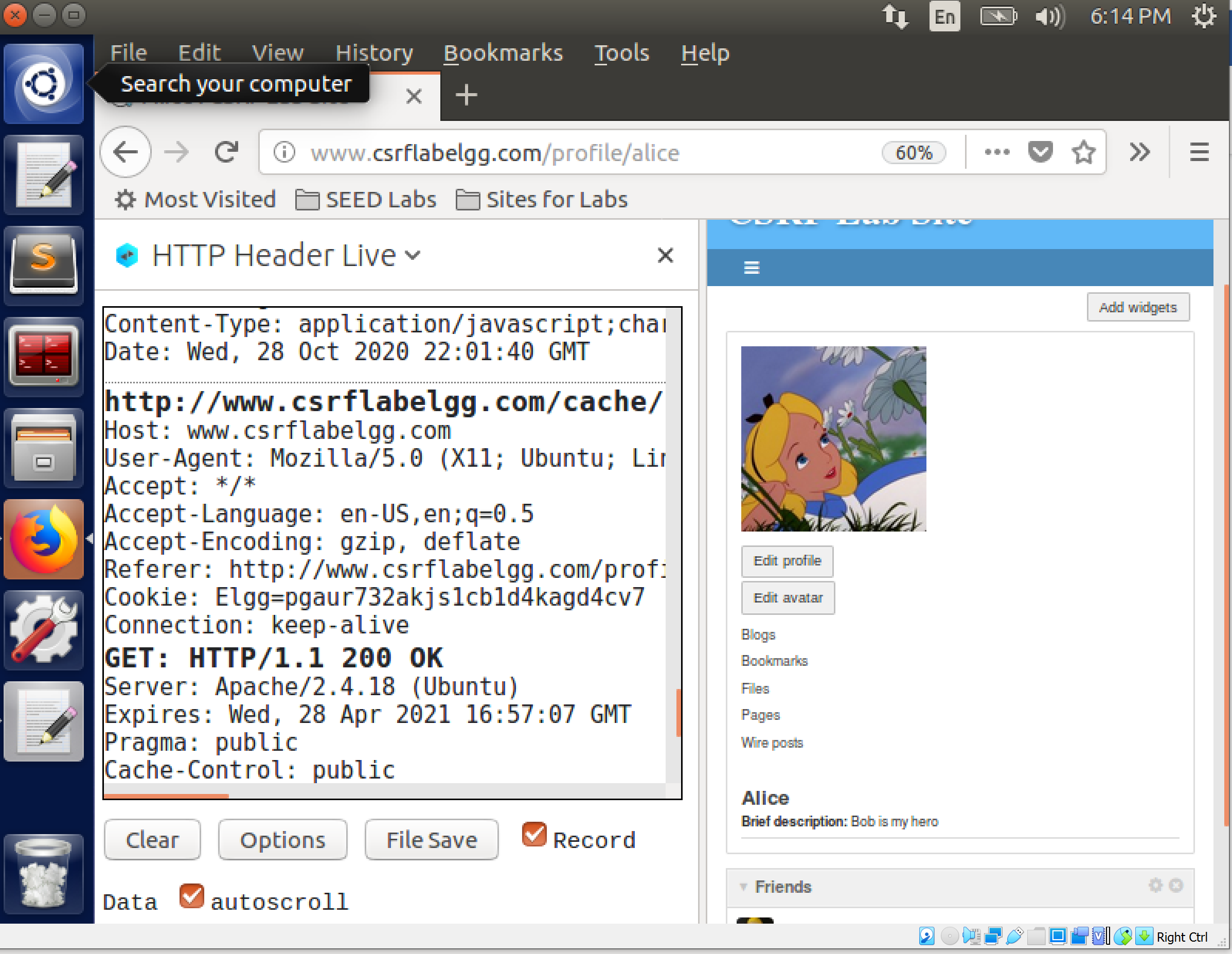
**Task 3: CSRF Attack using POST Request**

* To edit Alice's profile, we have to use the post request method.
* We first log into bob's account and select the edit account option.
* In the http request we observe that all the fields are visible to the public.
* We will need Alice's GUID and will need to write it to the brief description parameter.
* We can see that the GUID of Alice is 42. Using this we construct a page names editProfile.html.
* The program is as shown.We assume Alice clicks the link and we can see a change in her profile description.









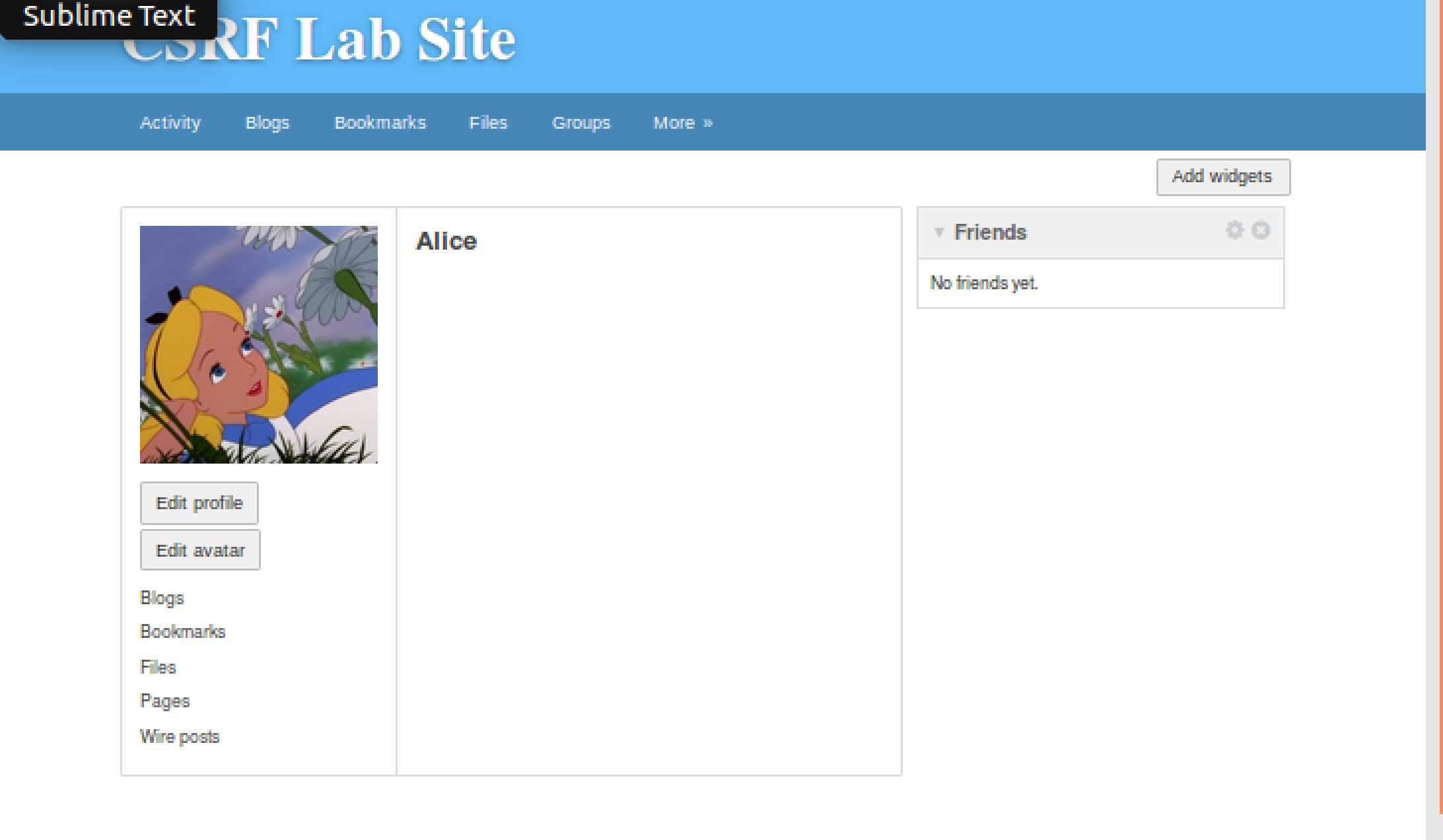
Thus our attack using the post method is successful.

**Question 1.**This particular problem can be solved similar to how we found Alice's GUID. We first search for alice on the website and then do an inspect element to view the GUID of the webpage owner. We don't need Alice's credentials to do this. Also we can try to just enter Alice's name as username and some random password and look at the HTTP REQUEST OR RESPONSE. This may sometimes contain Alice's GUID.

**Question 2.** In this case boby would not be successful in launching the attack because his webpage is different from that of the targeted website. The possibility of deriving the GUID is low as we don't have the website information. The GUID is sent only to the server of the targeted and not other servers of other websites, thus we would not get the GUID from the HTTP request from elgg to attackers’ websites

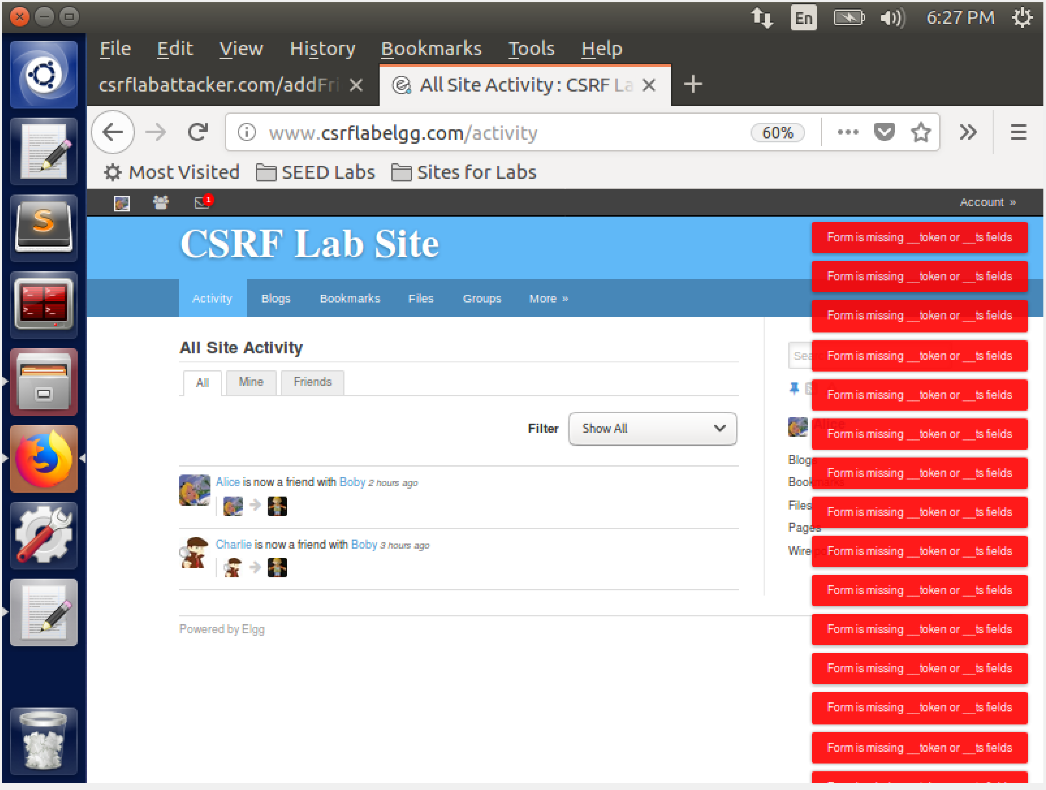
**Task 4: Implementing a countermeasure for Elgg**

We now enable the CSRF counter measure by commenting the true statement in the code as shown. We reset the account of Alice. We also see that the Elgg security token is a hash value got from the timestamp,user session if, and a random generated string.

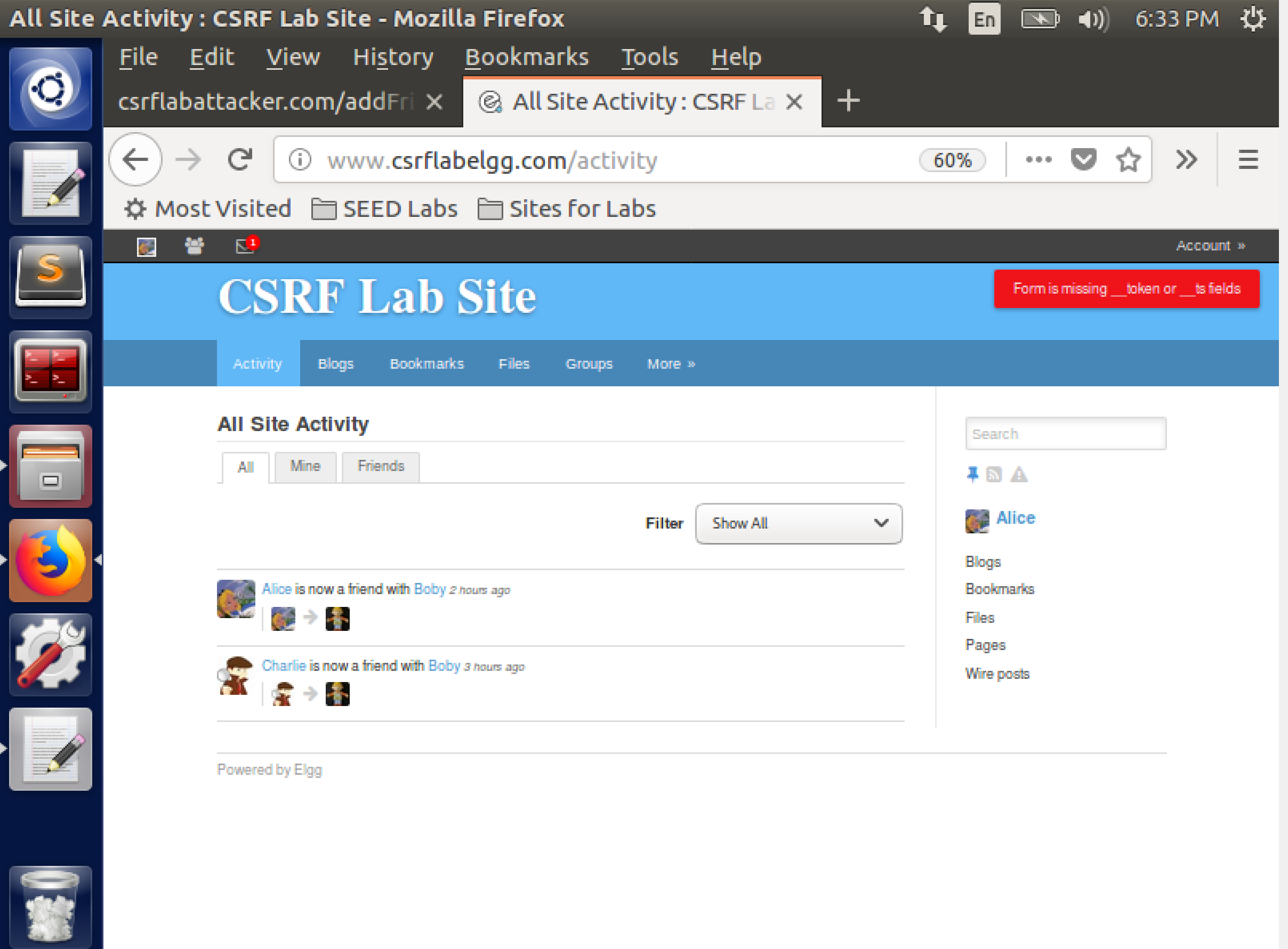




On executing the attack with POST method:



On executing the attack using the GET method:



We see that both times for post and get the attacks are unsuccessful. On looking at the header we see that no token and ts fields are being sent. This is because of the construction of the http request without specifying the timestamp and secret token.

**References:**

https://github.com/MeghaJakhotia/ComputerSecurityAttacks/blob/master/CSRF/CSRFLab.pdf