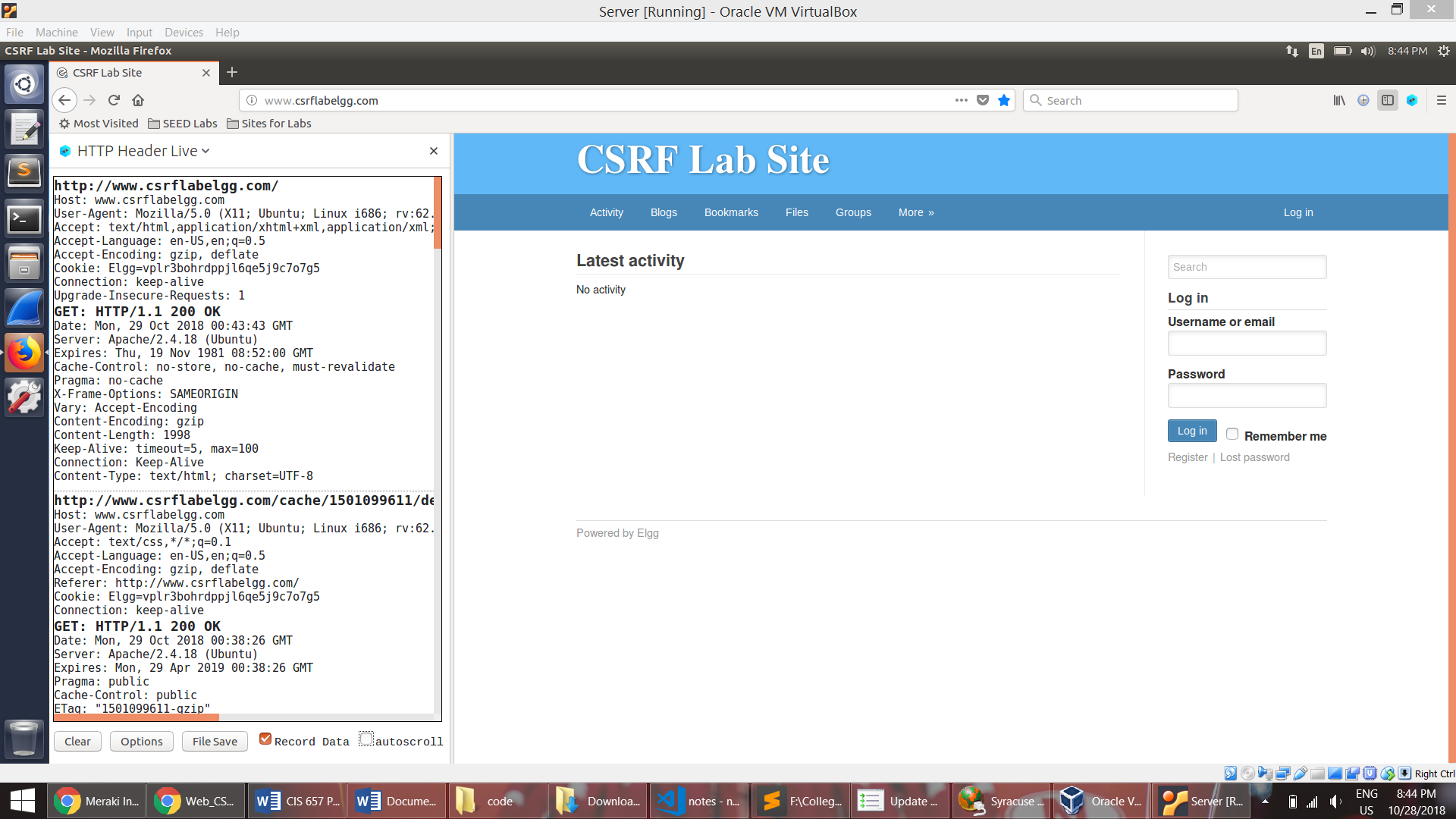
**Cross Site Request Forgery Lab**

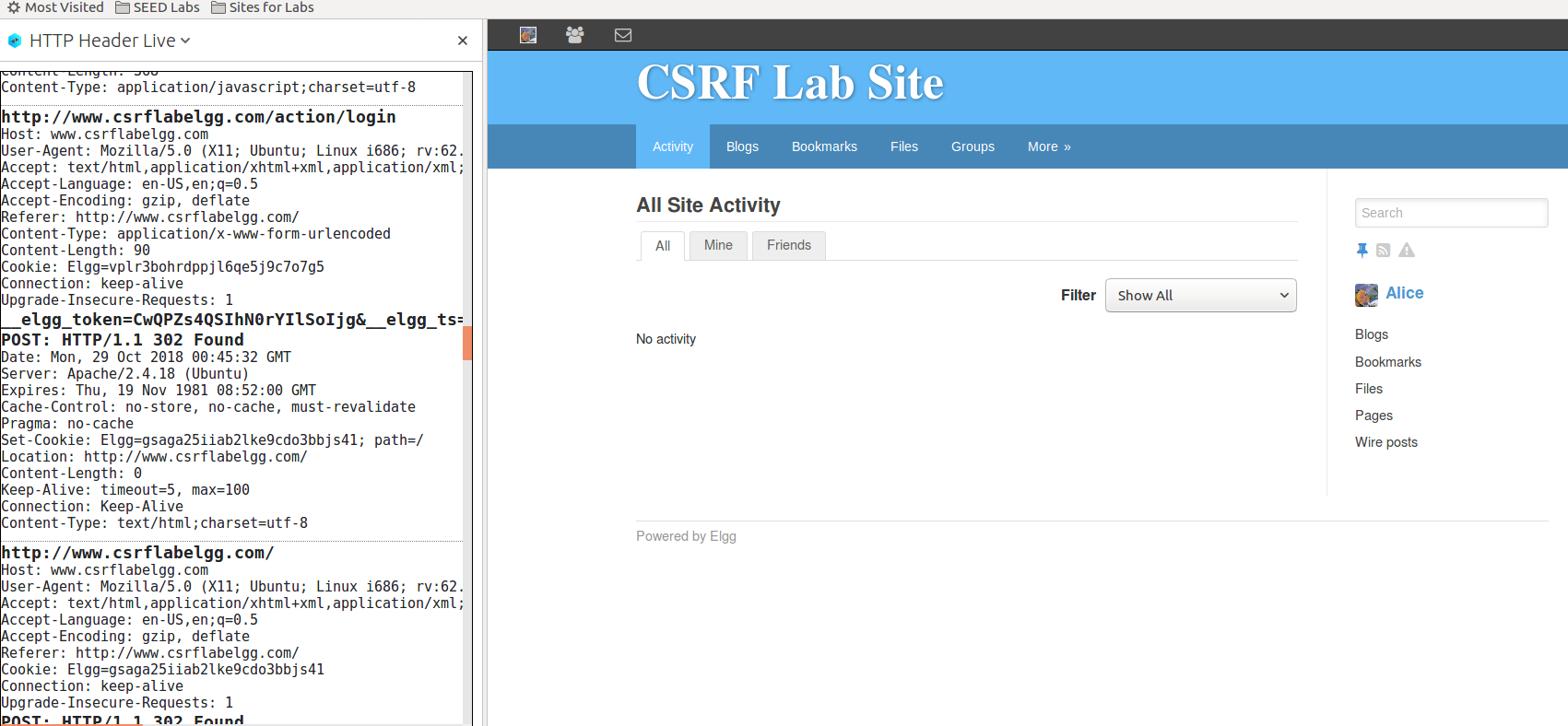
**Karan Amrutesh**

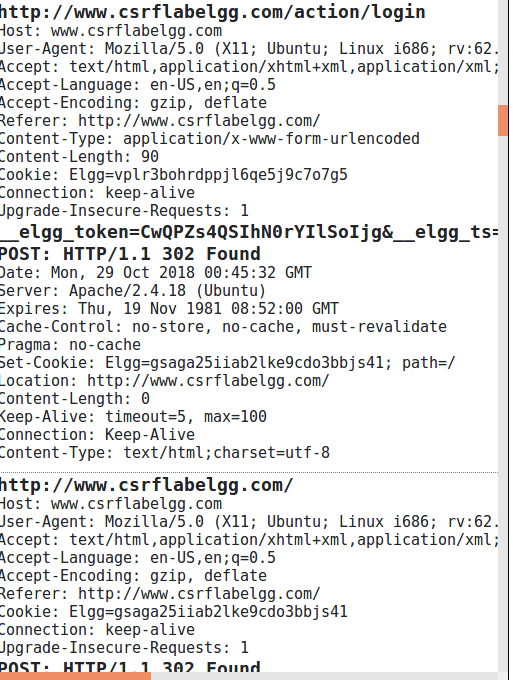
**Task 1: Observing HTTP Request:**

* We can see that the GET request does not have any parameters when we go the website.

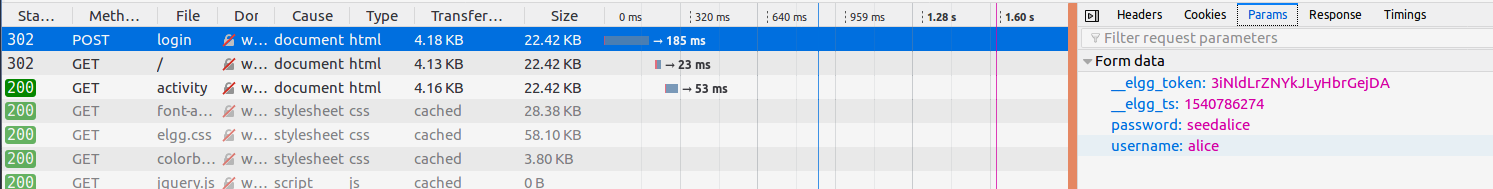


* When we login, it sends out a POST request. We can see that the user name and password are passed as parameters along with two parameters token \_\_elgg\_ts and \_\_elgg\_token.



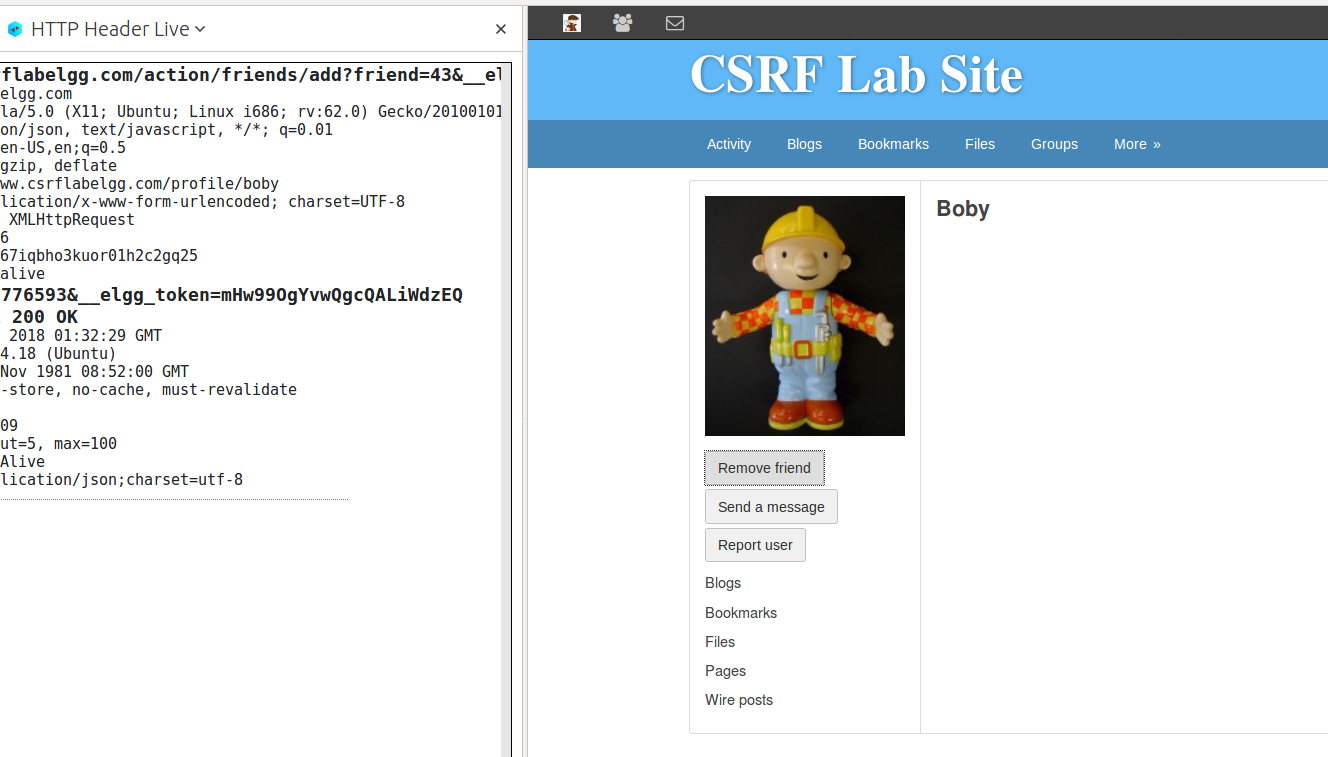
 

* Using the web developer tool to see the parameters in the POST request:

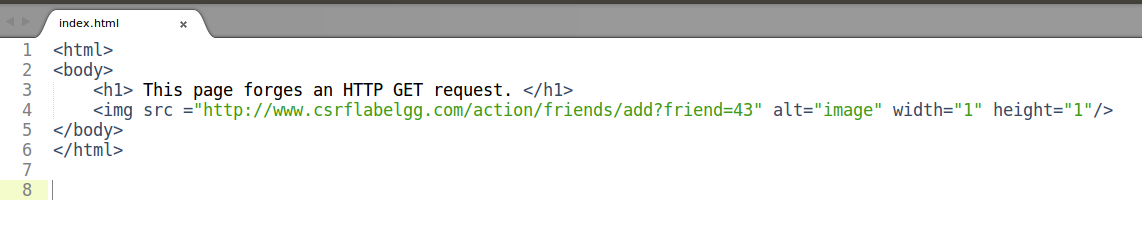


**3.2 Task 2: CSRF Attack using GET Request**

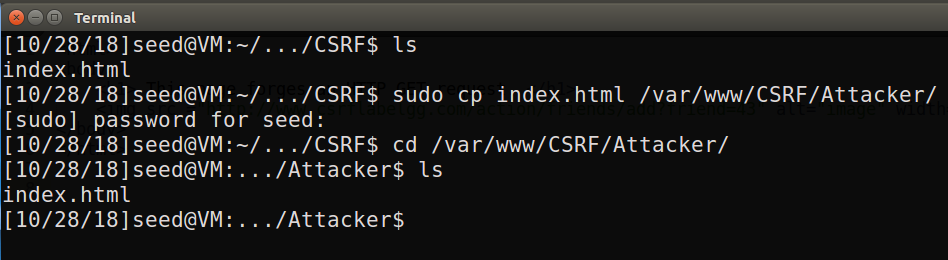
* We first add Boby as a friend to Charlie. When Charlie clicks ‘add friend’ button to add boby, we capture the request as shown below:



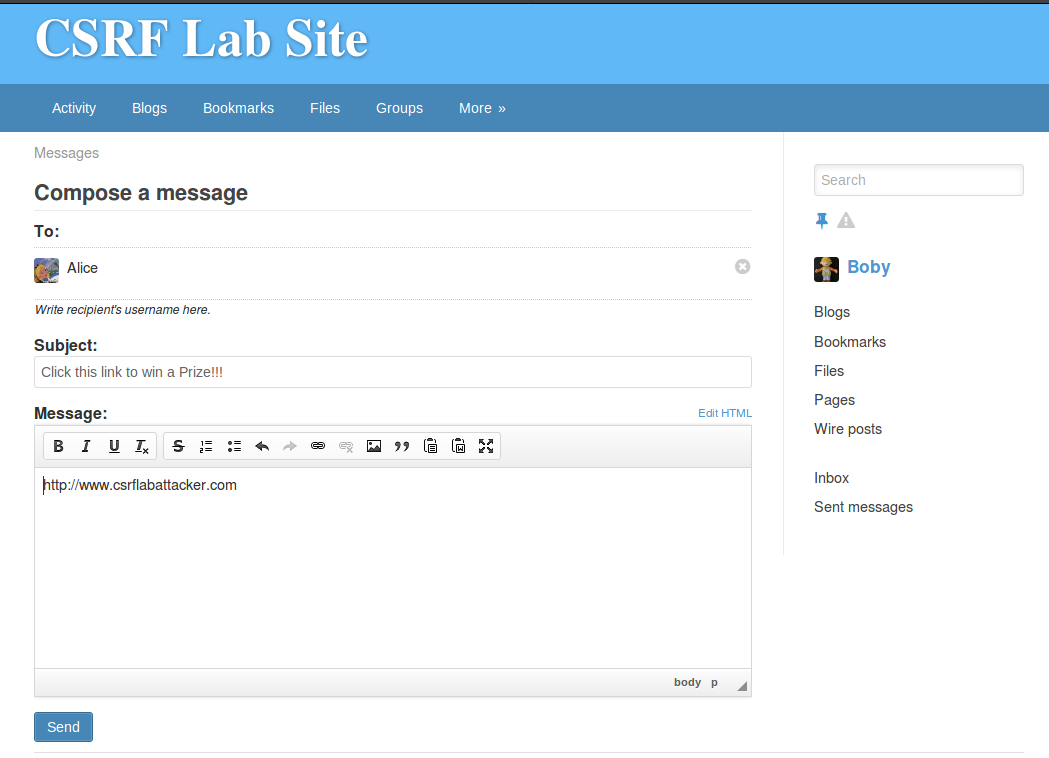
* From the request, we can see that the friend parameter is passed which indicates the user ID of Boby. So we get the GUID of Boby as 43.
* We also note the URL needed to add a friend. We use these information to construct our forge request:



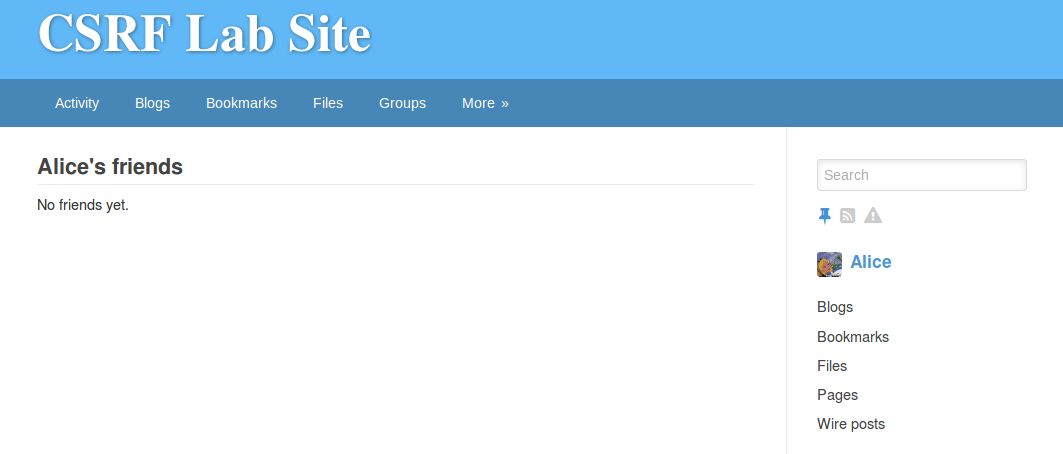
* The img tag is used as it will automatically trigger the HTTP GET request.



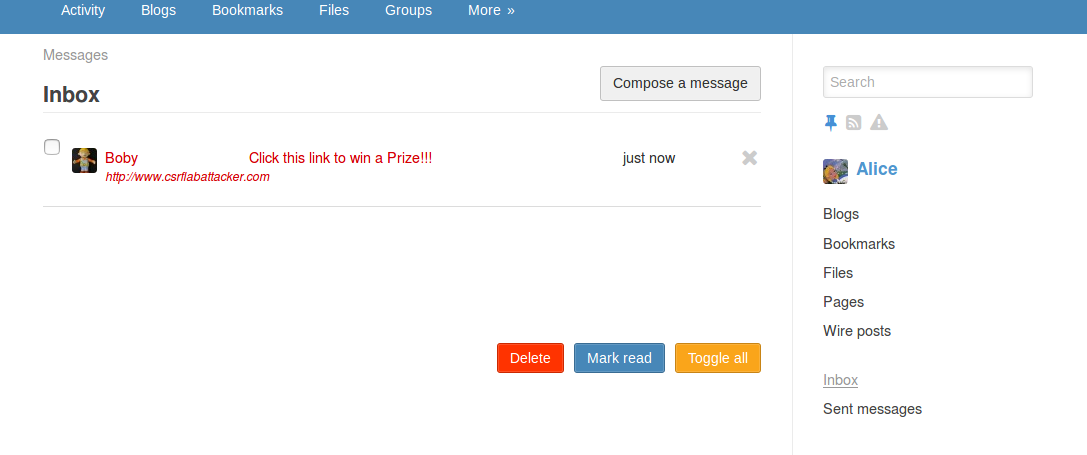
* We put this html file into the right folder so that when the link <http://www.csrflabattacker.com> is clicked, this file gets loaded and adds a friend.
* We then compose a message in Boby’s profile and send to Alice.



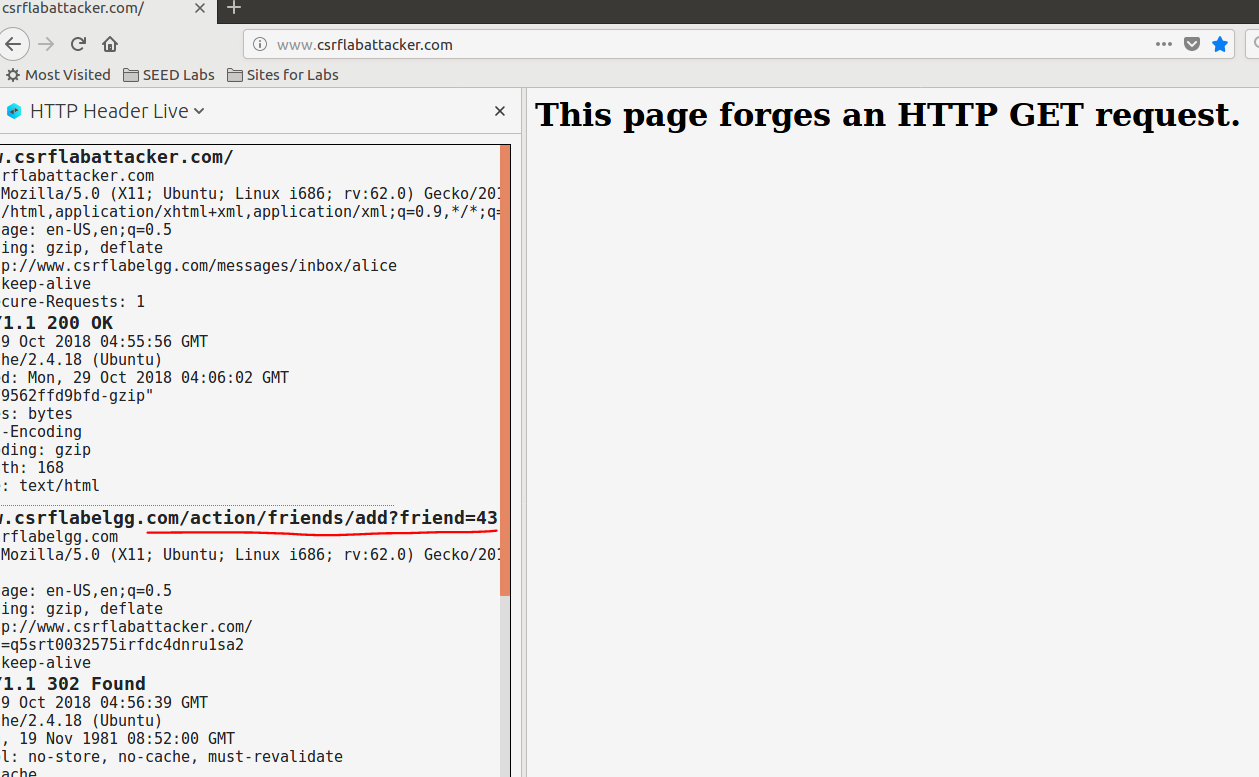
* We can see that Alice does not have any friends yet:

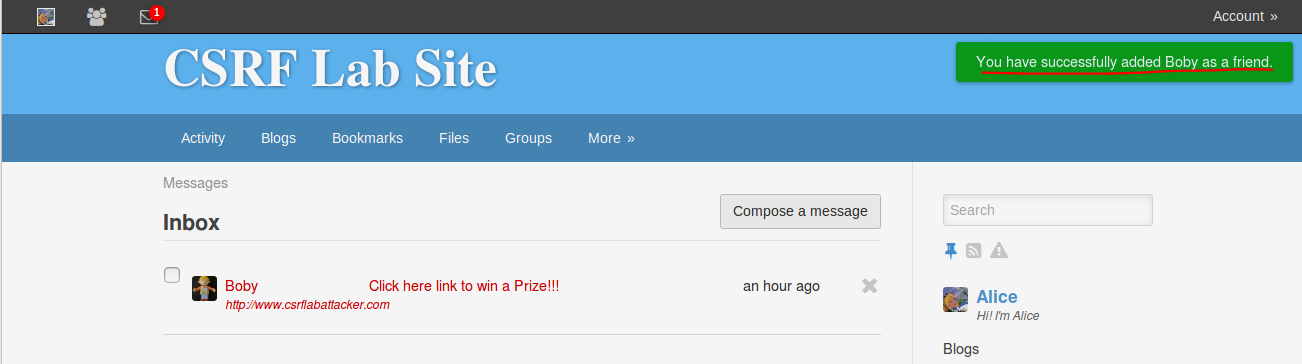


* The message sent by Boby:

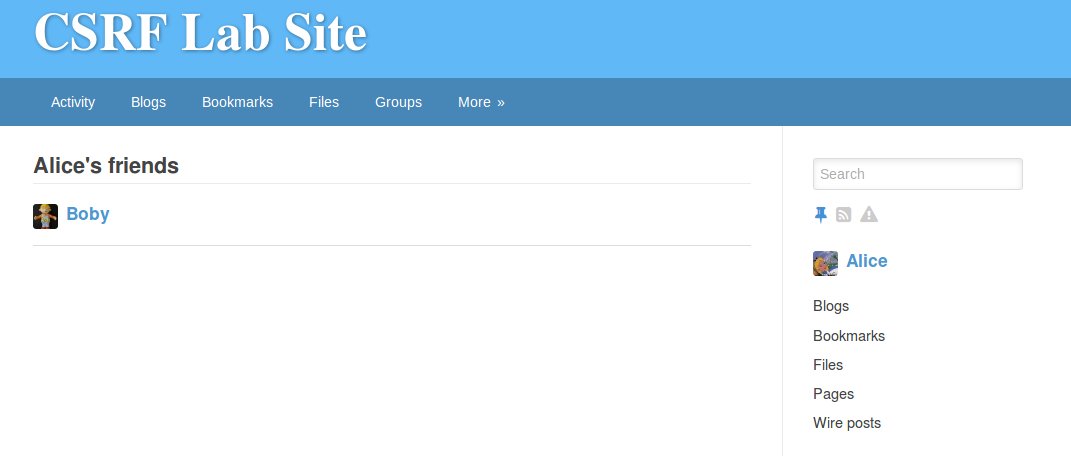


* When Alice clicks on the link, the attacker’s page is loaded and a GET request is sent to the csrflabelgg.com with the target URL mentioned in the attack’s html file, along with the friend parameter (friend=43) which add’s Boby as a friend to Alice.





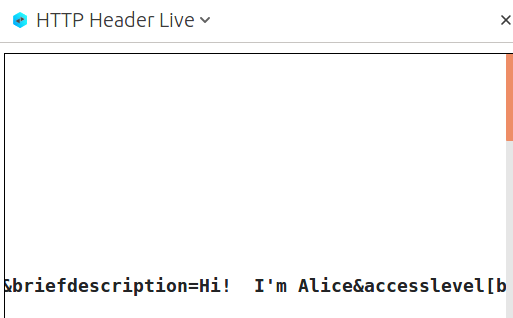
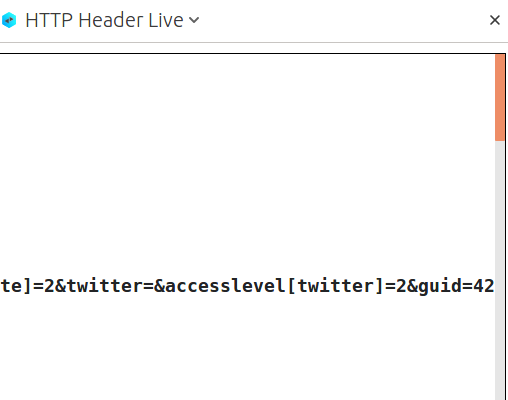
* Alice’s friend list after clicking on the link:



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

* We first click on the edit profile and submit the form by giving some description for Alice. We observe the parameters and get the GUID of Alice to be 42.

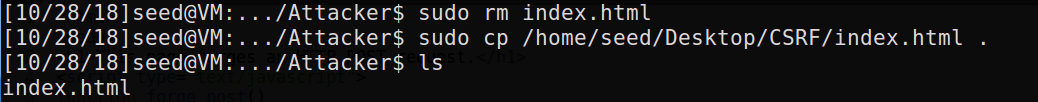


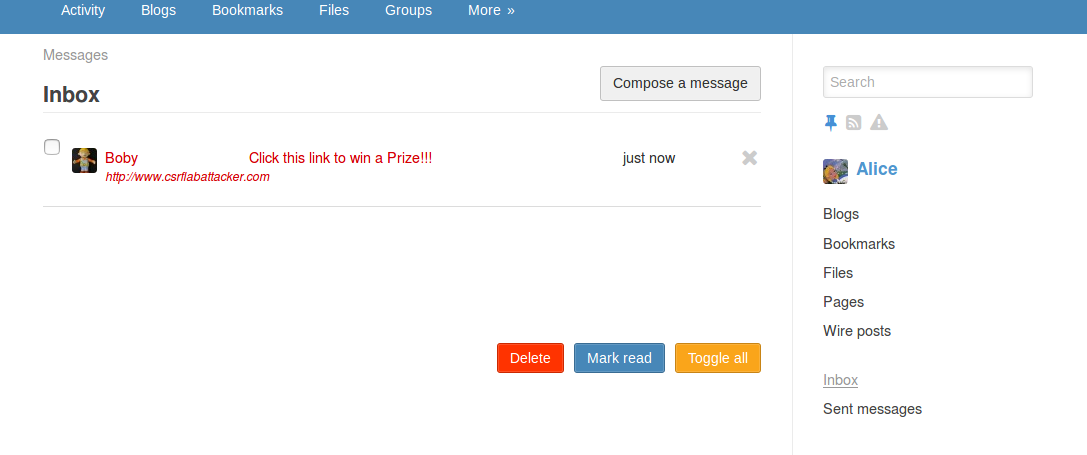
* We use these parameters to construct our target URL
* We use this URL to construct our html file which submits the form constructed, automatically on loading the page:



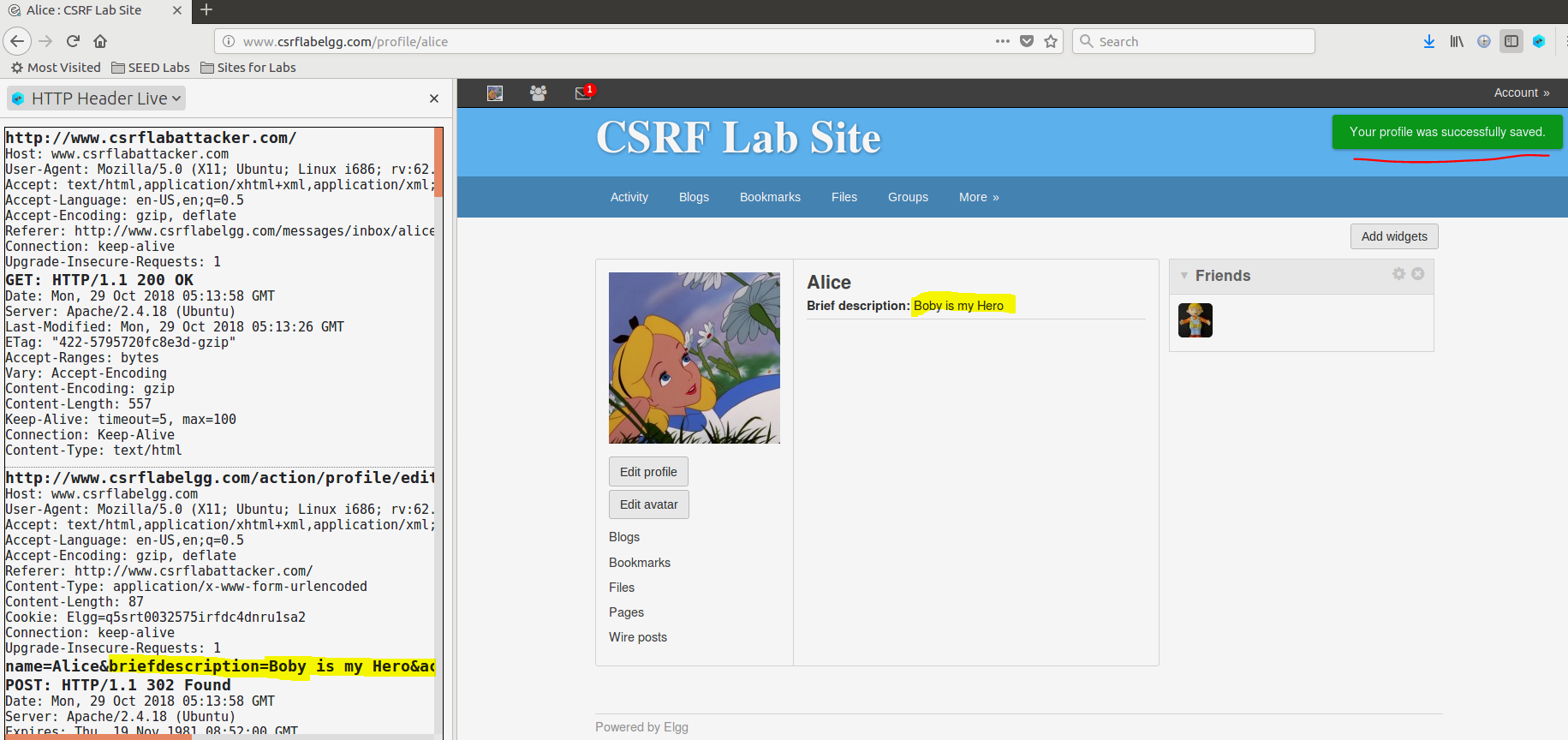
* Here, the name is Alice because we need to change Alice’s profile.
* briefDescription is the description we want to forge for Alice, which is Boby is my Hero
* accesslevel indicates that everyone can view this field
* GUID is Alice’s ID which we found out earlier.



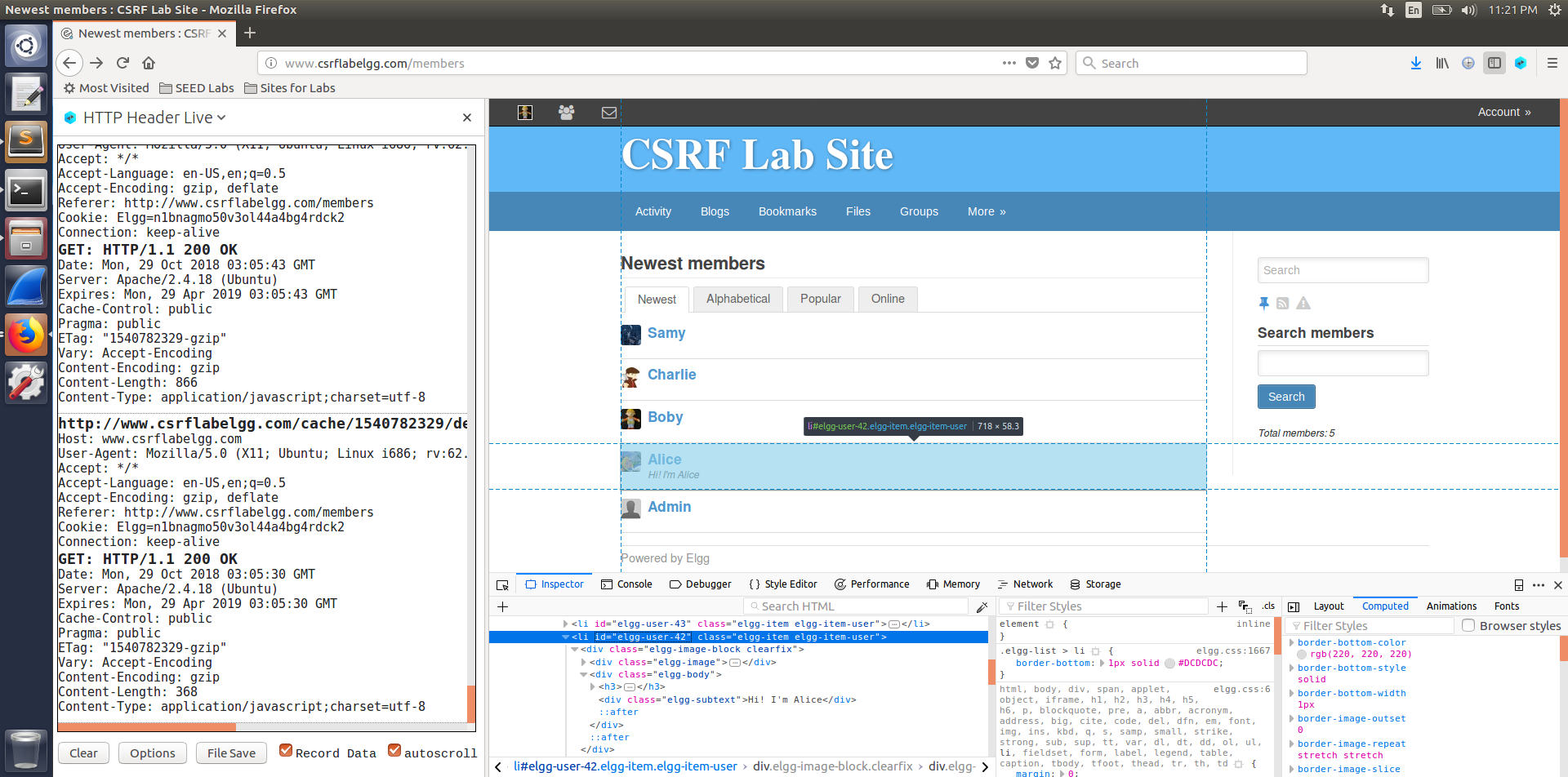
* Alice clicks on the link that Boby sent:



* When Alice clicks on link, our page is loaded and send the form we created back to the Alice’s profile using POST request. This causes the description to be changed
* We can see that the description in Alice’s profile has successfully changed:



* **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.
  + We can get the GUID of Alice by just viewing the member’s page and then inspecting the element which says “Alice” as shown below:

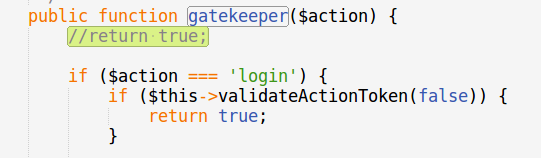


**Question 2:** If Boby would like to launch the attack to anybody who visits his malicious web page. In this case, he does not know who is visiting the web page beforehand. Can he still launch the CSRF attack to modify the victim’s Elgg profile? Please explain.

* No we cannot launch the CSRF attack as we need to have the GUID of the victim. Otherwise we would not know to construct the target URL to forge the request.

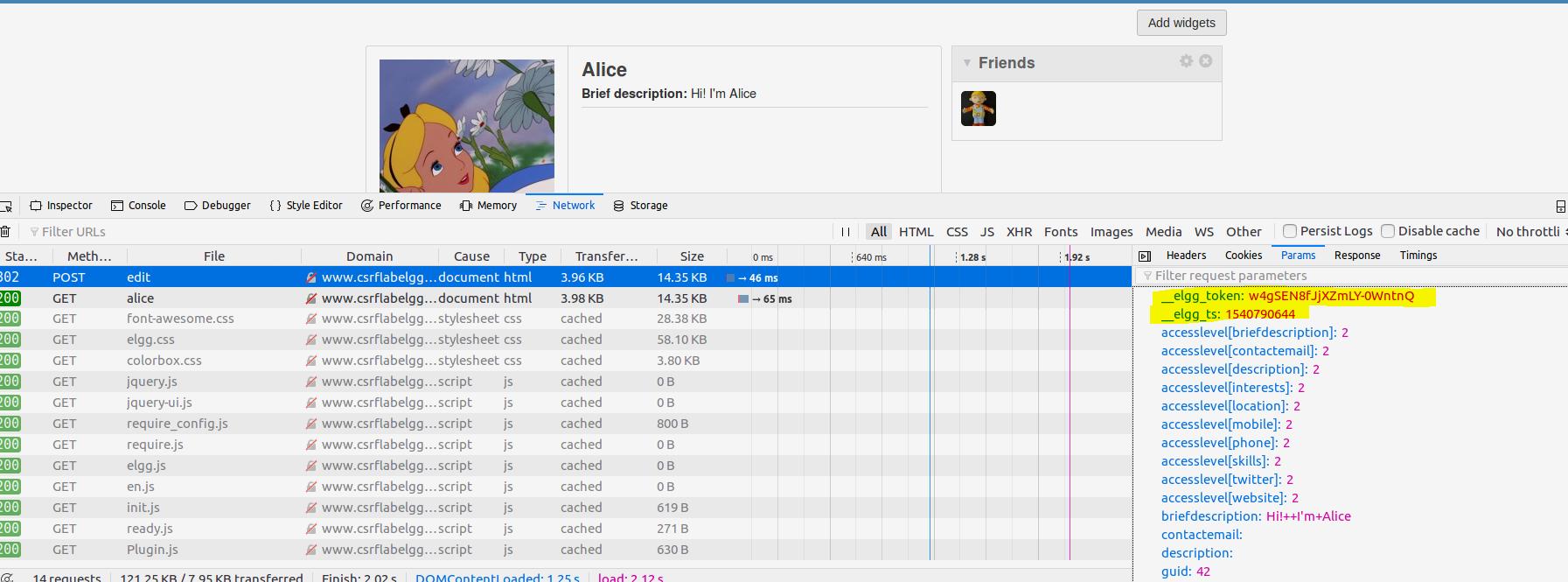
**Task 3: Implementing a countermeasure for Elgg**:

* We comment out the return statement:

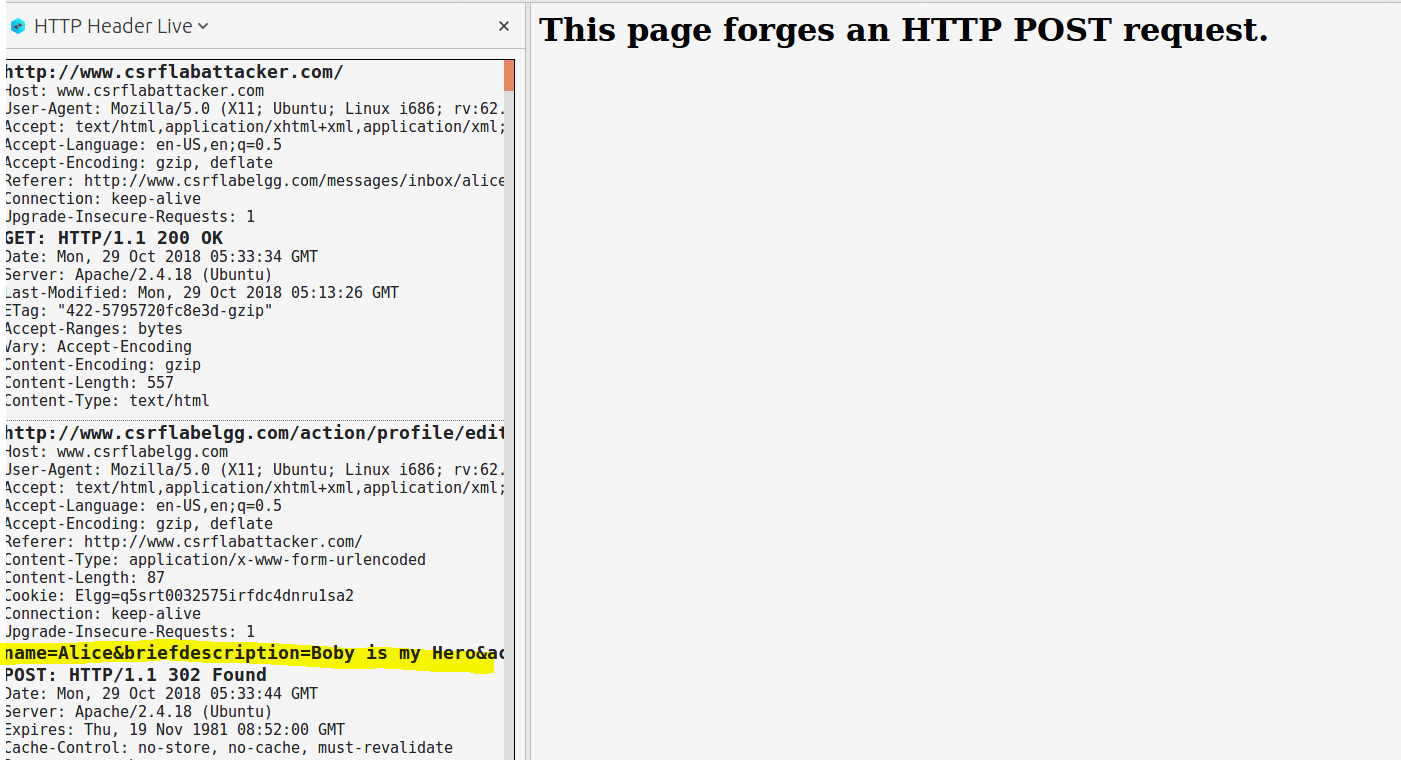


**Please point out the secret tokens in the HTTP request:**

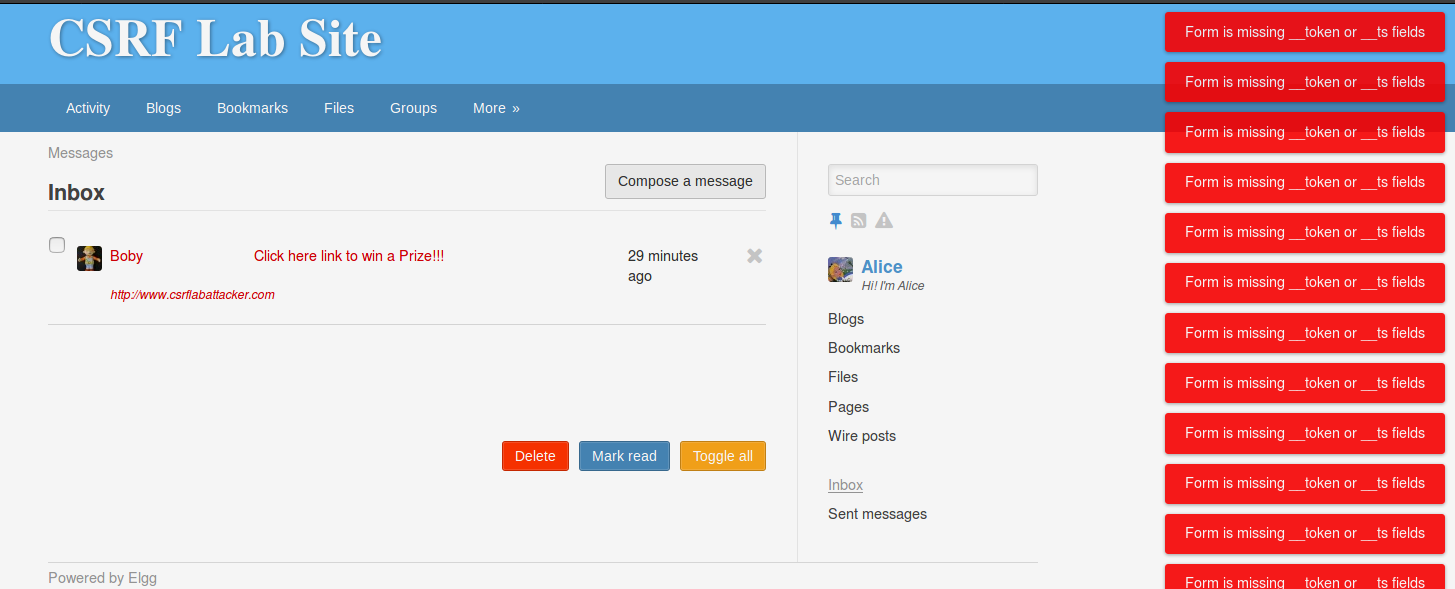
* When we edit Alice’s profile, we can see the parameters passed in POST request:
* We can see that the secret token parameters (\_\_elgg\_token and \_\_elgg\_ts) are passed along with the other parameters:



* When we try to edit Alice’s profile by clicking on the link sent by Boby like before, the attack fails.
* This is because, we need to know the tokens to perform the action. Since the token is not present, the validation fails and the action will be denied.



* We can see the error saying form is missing the secret token fields:



* The attackers cannot get the value of this secret token as it is an MD5 hash value of secret value, timestamp, user session ID and a randomly generated session string. Hence this is hard to guess by the attackers.