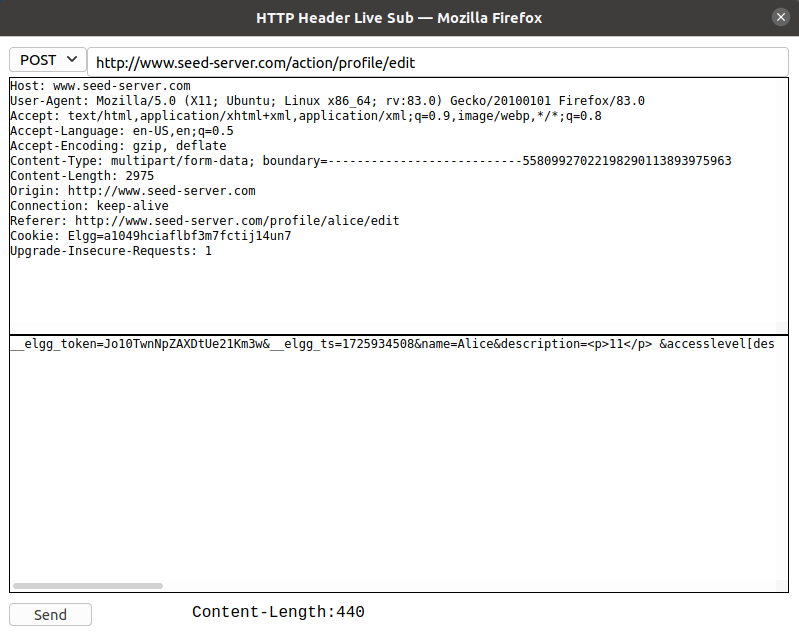
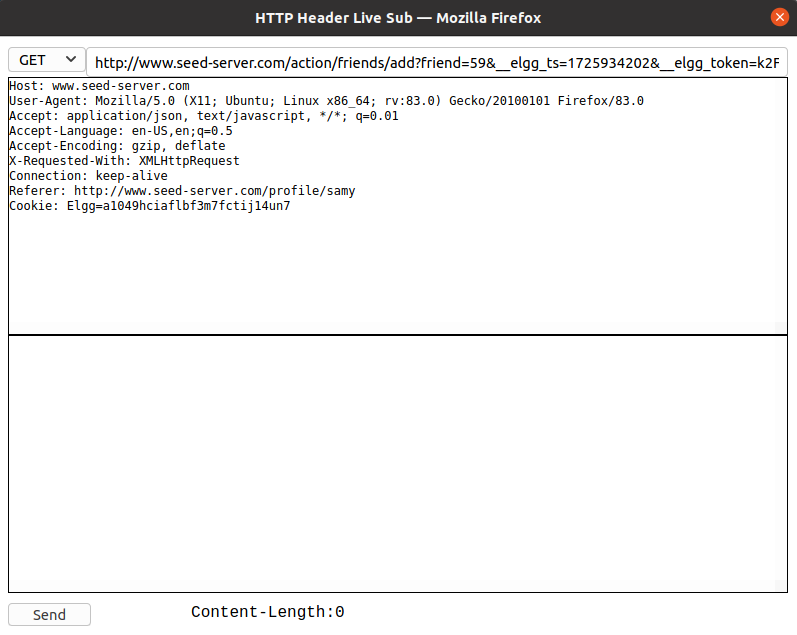
Name: Donghao Li  
SUID: dli106

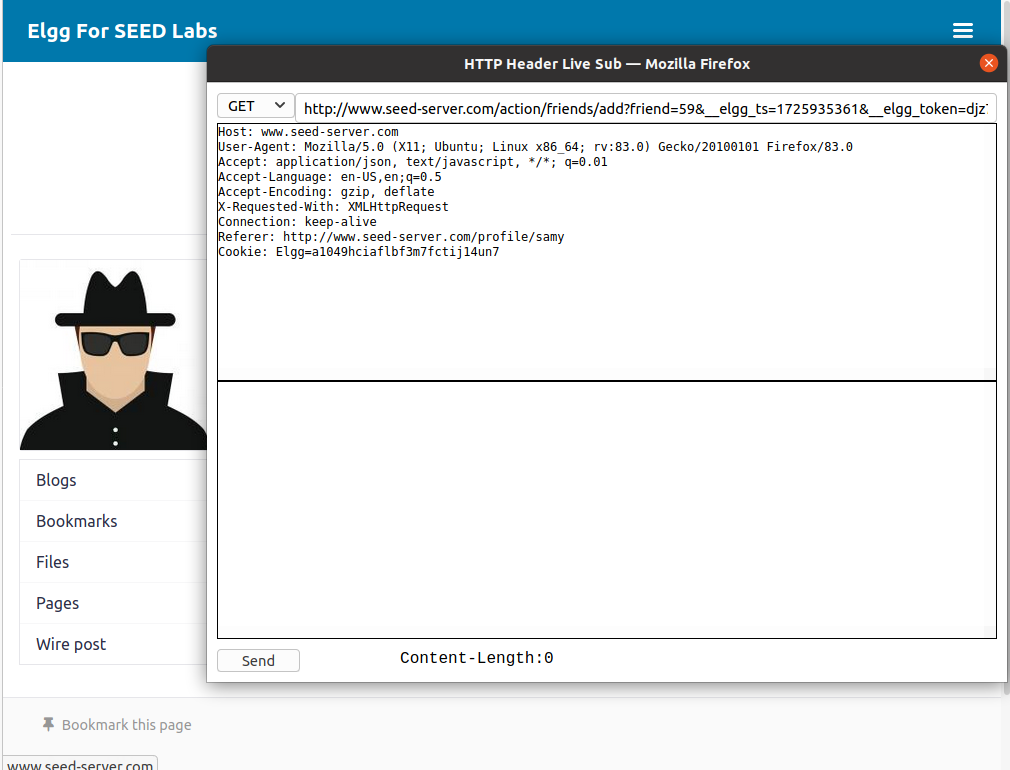
Task 1: Observing HTTP Request.



**Answer: As in the first figure, it is a get request from http. It describes the host and the user's action which is add friend.**

**As in the second figure, it is a post request from http. It shows the user Alice making an update in the about me part. The user Alice added a 1 in the about me part.**

Task 2: CSRF Attack using GET Request

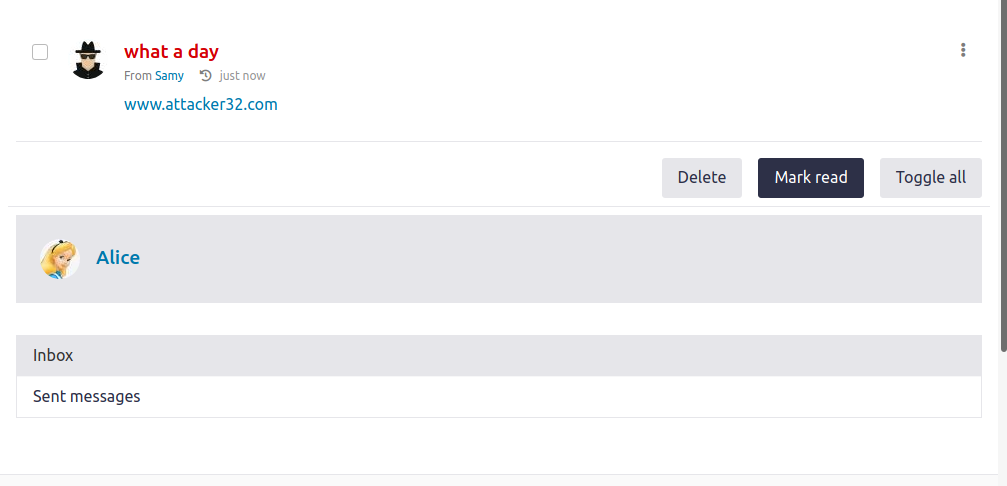


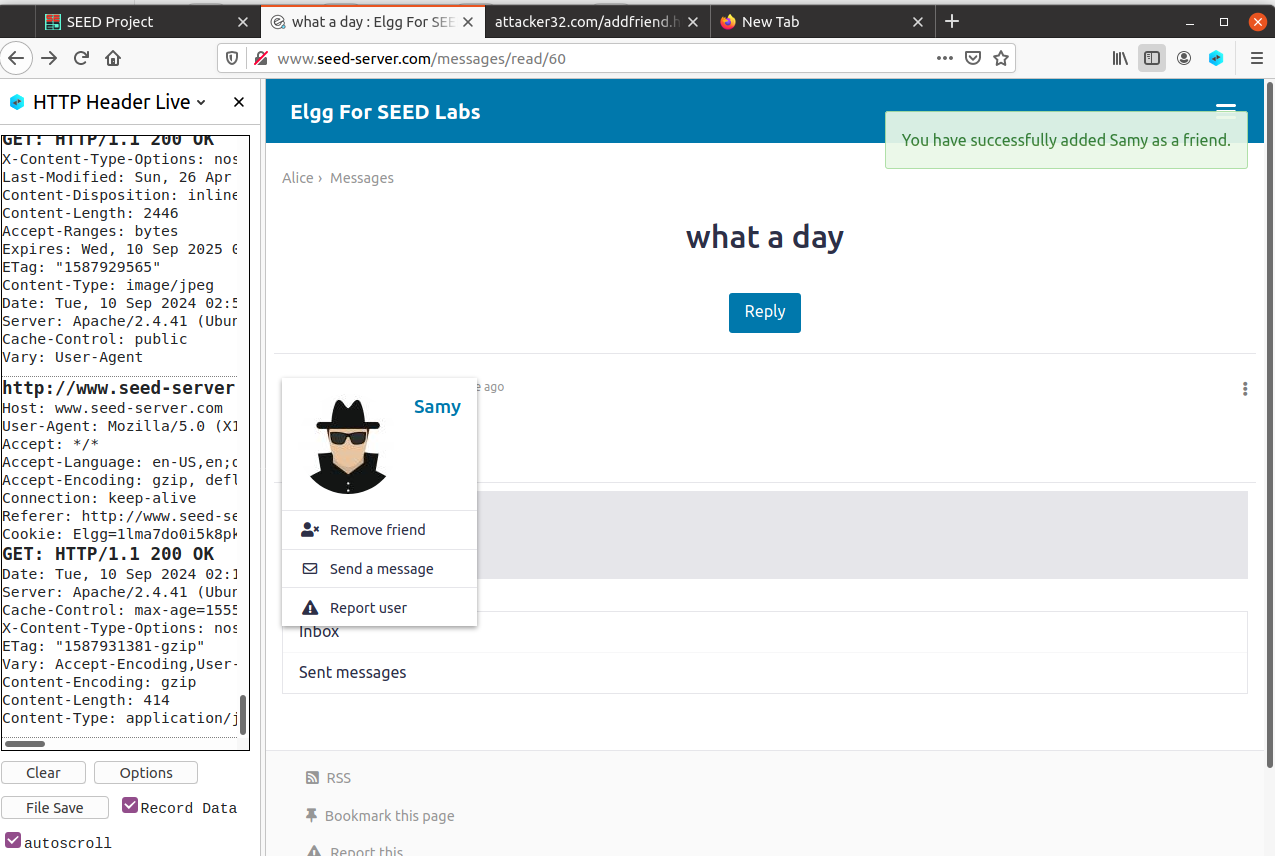
**Answer: In figure one, we can find the friend parameter for Samy is 59. As well as the format of adding a friend in the list.**

**“**[**http://www.seed-server.com/action/friends/add?friend=59&\_\_elgg\_ts=1725935361&\_\_elgg\_token=djz7**](http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1725935361&__elgg_token=djz7)**”**

**since we can dismiss ts and token, the edited code for addfriend.html should look like this**  

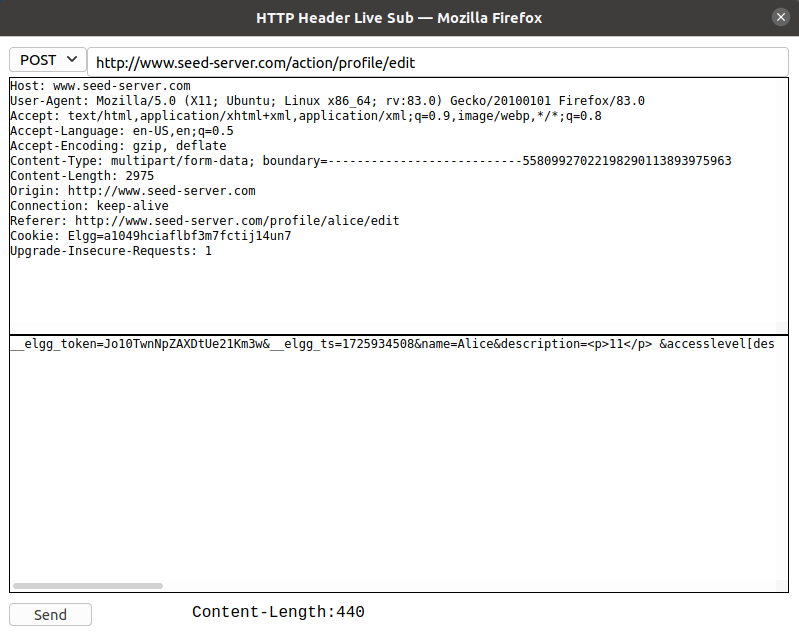

**once Alice clicks on the URL, it will lead Alice to the addfriend.html and it will automatically run the add friend request for Alice, and the target of this add friend request is Samy.**





**send the attack link to Alice through Samy’s file, once Alice clicks on the attacker32.com and further goes to addfriend attack, Samy is automatically added to Alice’s friend list.**

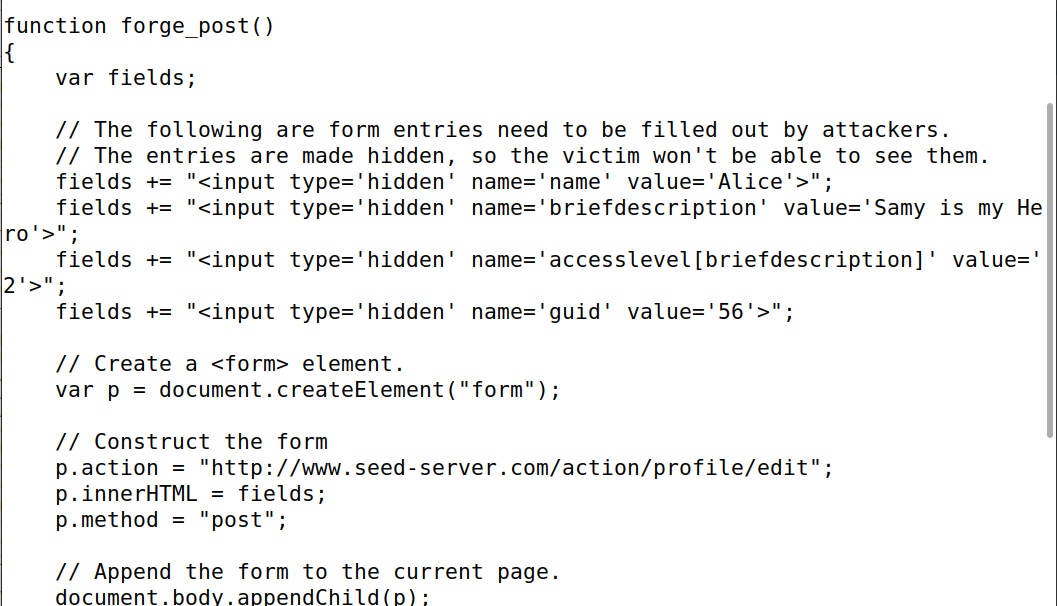
Task 3: CSRF Attack using POST Request



**Answer: By edit Alice’s profile we can get a post request and figure out the following information.**

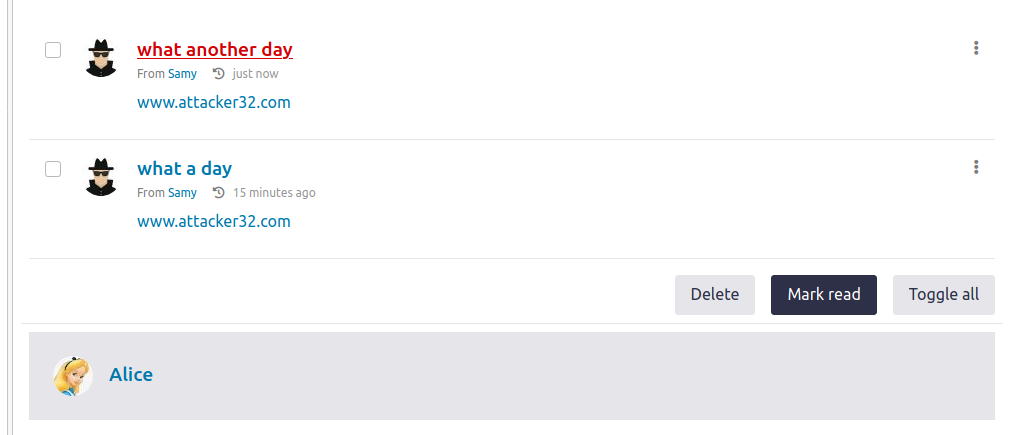
**Name is Alice, the description is empty, accesslevel is 2, and guid is 56.**

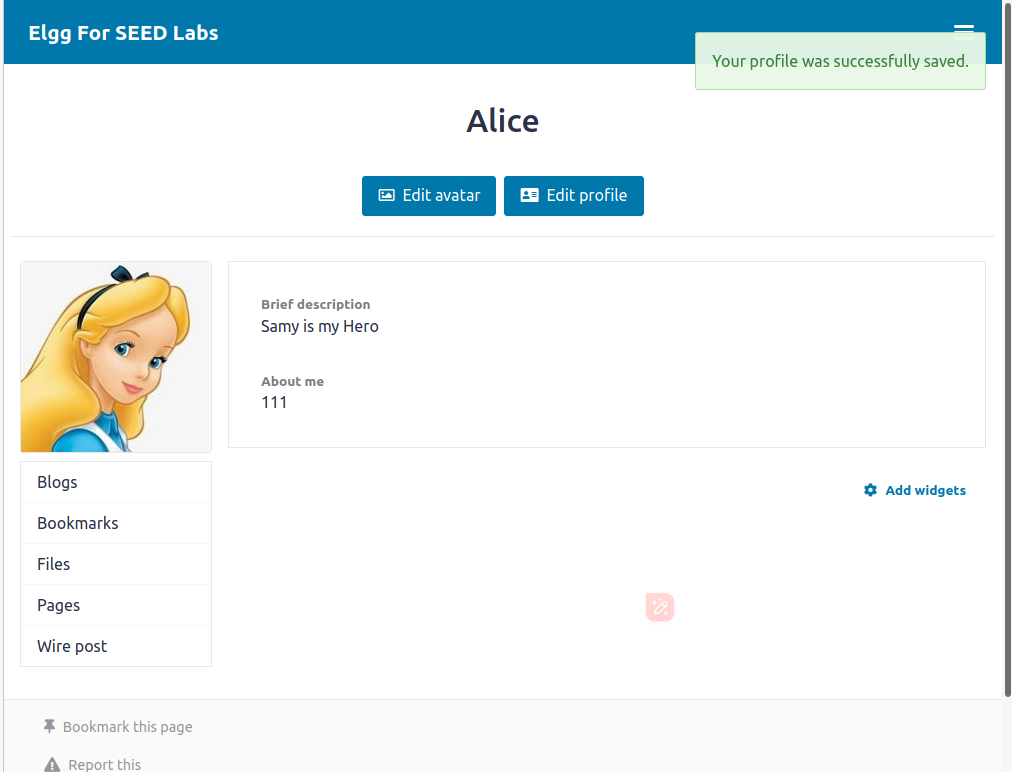
**“\_\_elgg\_token=WNsyLP8t7lQEHZw4M1QPdg&\_\_elgg\_ts=1725937195&name=Alice&description=<p>111</p> &accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=56”**



**Then as shown in the figure, change the p.action address to edit profile address shown in the post request.**

**Therefore, the code for editprofile.html should be this. By writing so, the phishing web page would send a post request as user Alice to elgg web page and change the description information.**



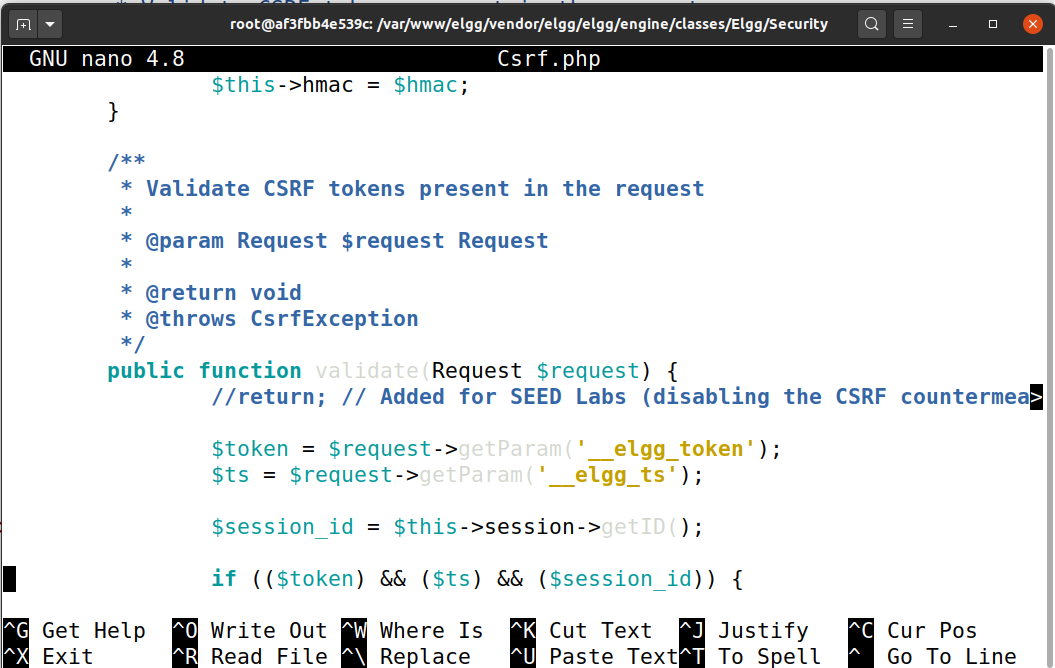


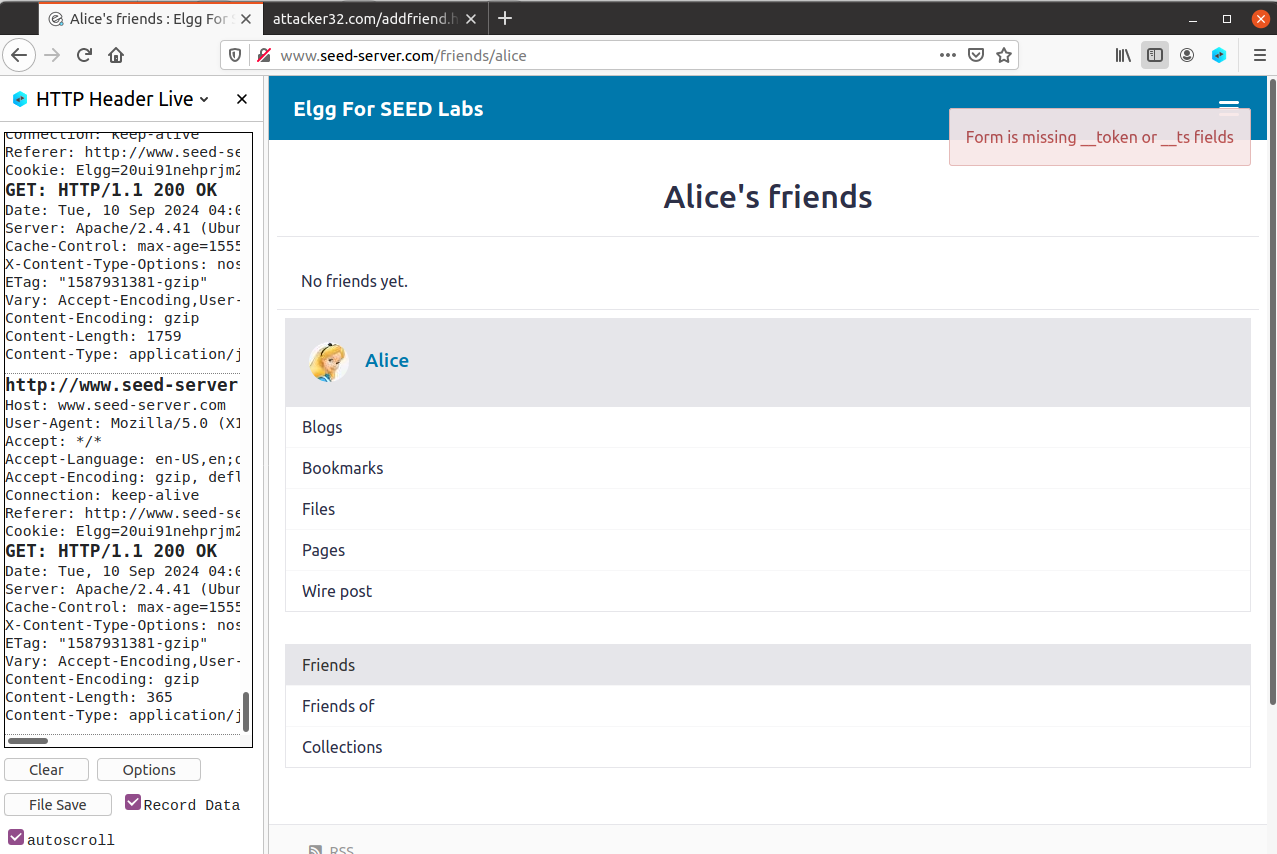
**As you can see in the figure, after Alice’s profile click attacker32.com link and goes to editprofile.html, the description has changed.**  
Question 1: The forged HTTP request needs Alice’s user id (guid) to work properly. If Boby targets Alice specifically, before the attack, he can find ways to 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.  
**Answer: Since Samy has already achieved an addfriend attack on Alice’s profile, he would be able to get enough information that’s needed for an edit profile attack including user name, accesslevel, as well as the guid.**

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.

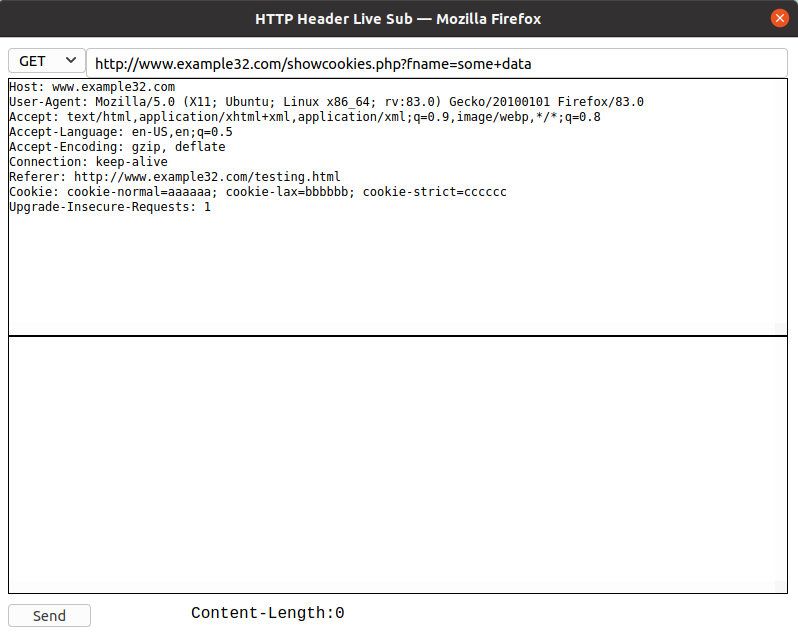
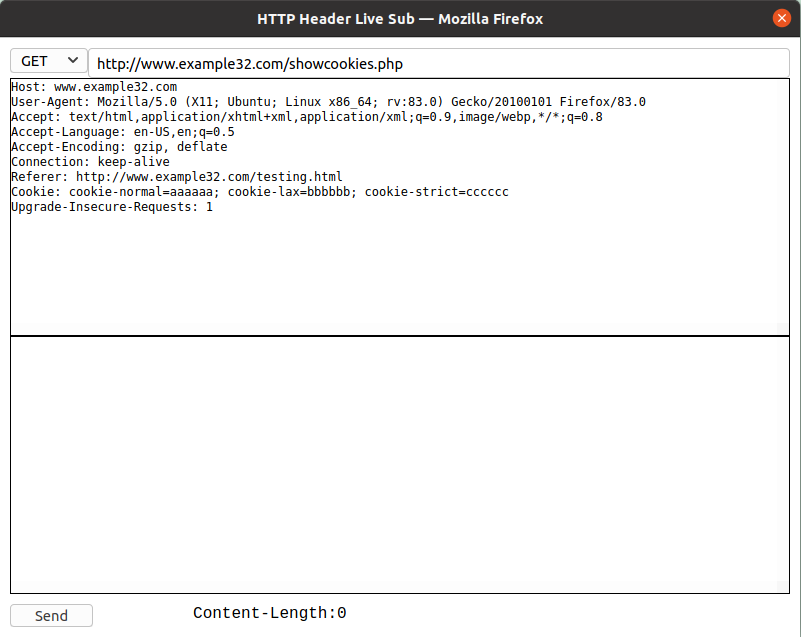
**Answer: That would probably work cause Samy has multiple chances to try out every guids he got.**

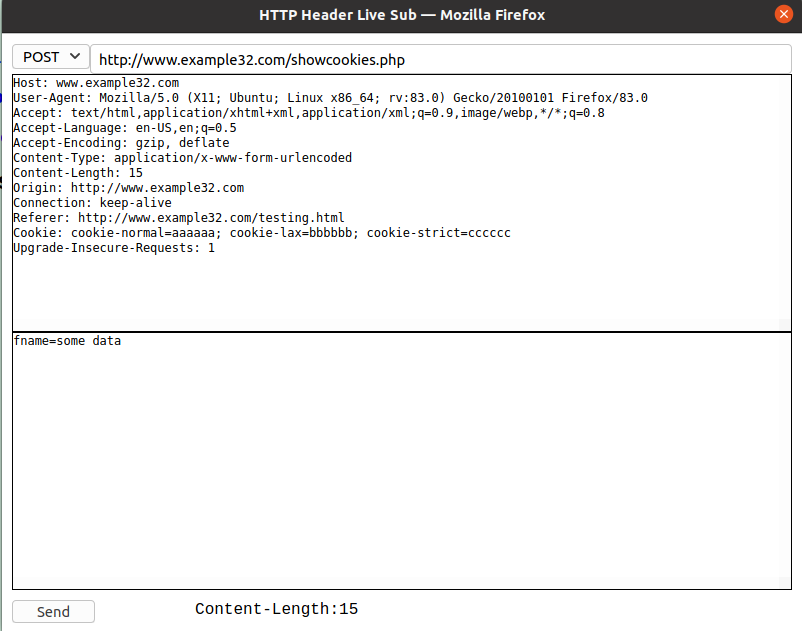
**Task 4: Enabling Elgg’s Countermeasure**

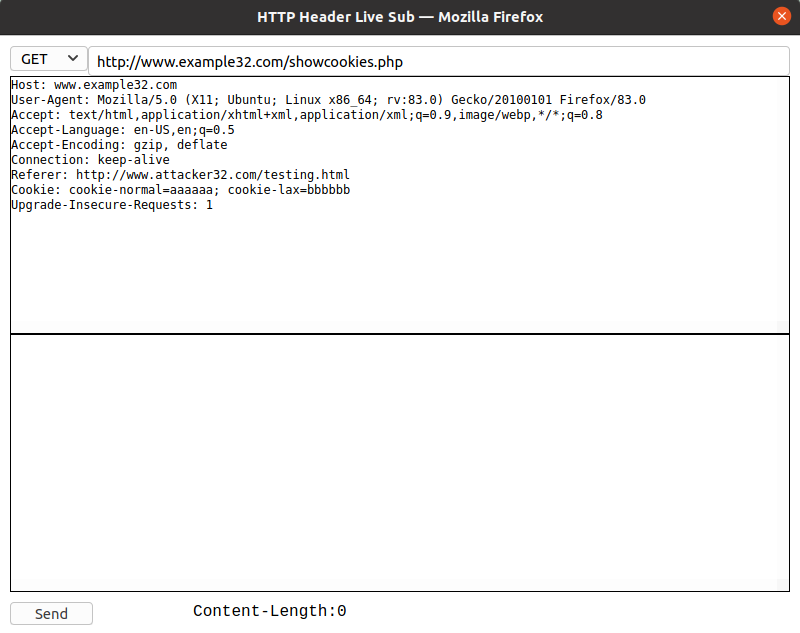


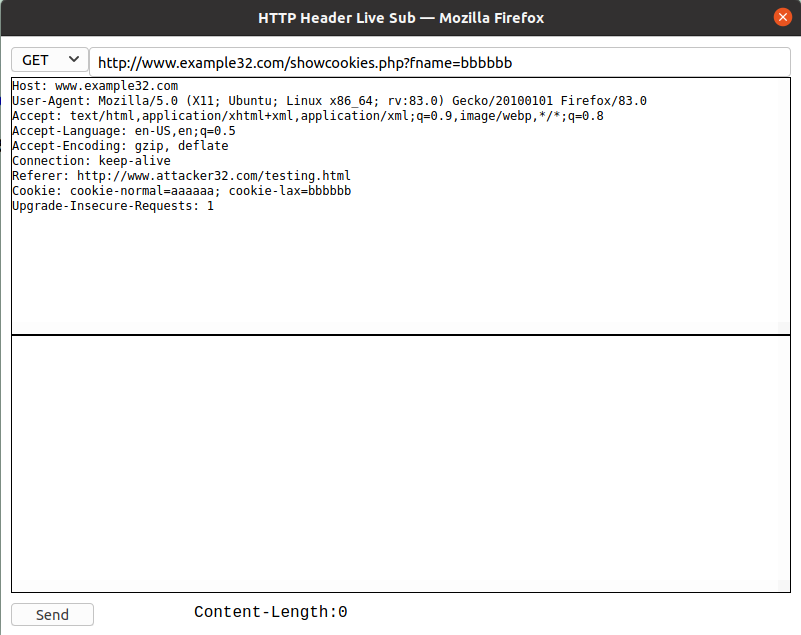


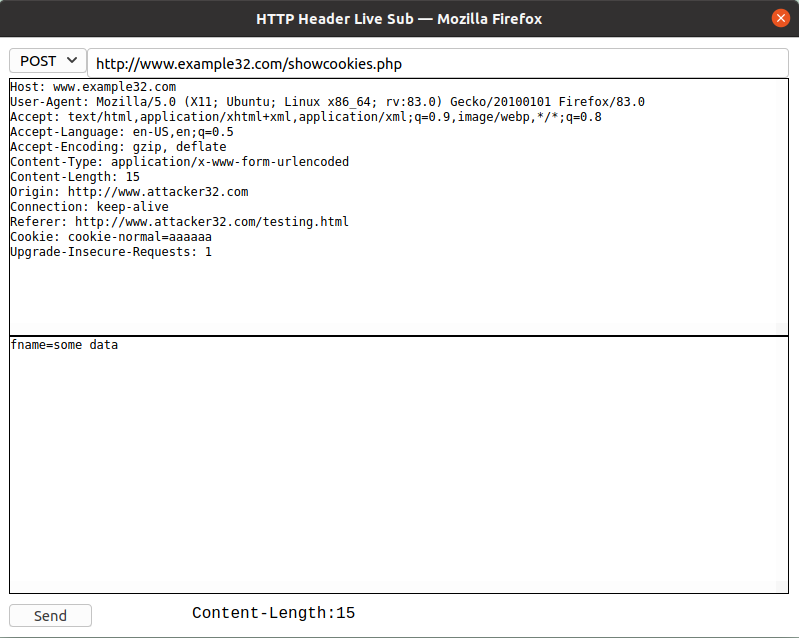
**After comment the return function inside the container, the addfriend attack and editprofile attack does not work any more.**











Task 5: Experimenting with the SameSite Cookie Method

Please describe what you see and explain why some cookies are not sent in certain scenarios.

**Answer: All three kinds of same site request contains three cookies. Where as for cross site requests, get request link and form only contain normal and lax cookie. post request form only contains normal cookie. By making sure strict only through same site requests, it decrease the risk of making important action from cross site request**

• Based on your understanding, please describe how the SameSite cookies can help a server detect whether a request is a cross-site or same-site request.

**Answer: When a server receive a request, with several cookies, it would be able to check whether the request contain lax and strict cookie, if so then it is a same site request, if not then it is a cross site request.**

• Please describe how you would use the SameSite cookie mechanism to help Elgg defend against CSRF attacks. You only need to describe general ideas, and there is no need to implement them.

**Answer:Set Session Cookies with SameSite=Lax or SameSite=Strict**

**Use SameSite=Strict for Critical Cookies:**

**Elgg can be designed to detect when a request is missing critical cookies (like SameSite=Strict cookies) and block or log these requests as potential CSRF attempts.**