Seunghyuk Baek

Sbaek44@vt.edu

ECE5480 Fall2018

Project 4

Objective

Cross-Site Request Forgery (CSRF) is delivery of malicious unauthorized requests. This delivery is often involving trusted web application, so the victim of the attack opens malicious message without suspicion. The consequence of this attack can be exposed user account to hackers. This report will examine steps of CSRF with scenario-based approach and basic depending mechanism for this security exploit.

Basic Mechanism of web application.

Web pages consist of front and back end codes that provide interface for a web application and process business logics. Users use web browsers to utilize services that the web application provide. This utilization is done by HTTP protocol.

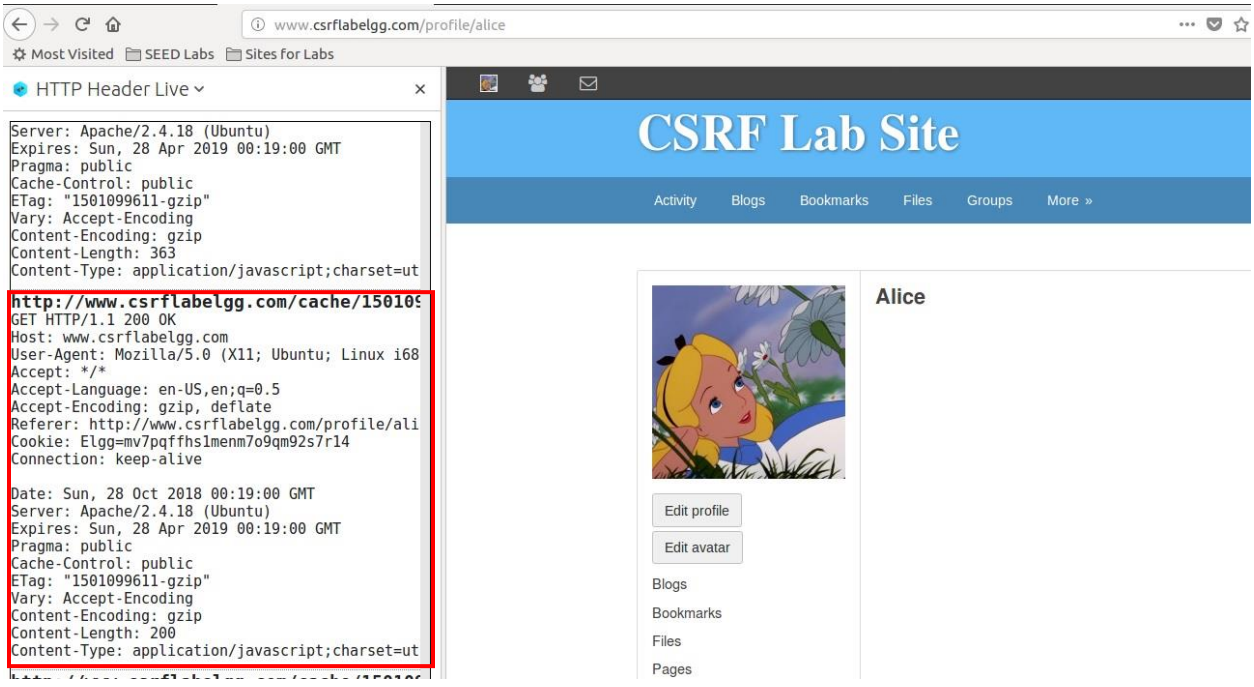

Figure 1. Log in as Alice and click the profile picture at top left corner.

Above figure 1 is demonstration of this HTTP protocol, the user Alice logs into csrflabelgg.com which is pseudo web application set up for this demonstration. Then, Alice clicks top left profile picture to see her profile for this website. When she clicks the profile picture, her Firefox browser sends HTTP GET request to the web server. Then the webserver responses with 200 OK code which means request is received and accepted. This is basic communication methods between a web server and users' web browsers. HTTP protocols have many other status codes. For example, 3XX codes are for redirection, 4XX for client error and so on. Along with the status codes, the request and response also carry information such as users' operating system, length of the content, dates and server.

Conducting CSRF attack.

Let's assume that Boby is malicious hacker, and he wants to add Alice in his friend list. In Elgg website, anyone can add another user to his or her friend list but both users have to agree on request to become mutual friends. To do this, Boby first adds Alice on his friend list. This friend request is passed as URL parameter, therefore, Boby can easily find out Alices' GUID which is the unique ID for Alice in this web application.

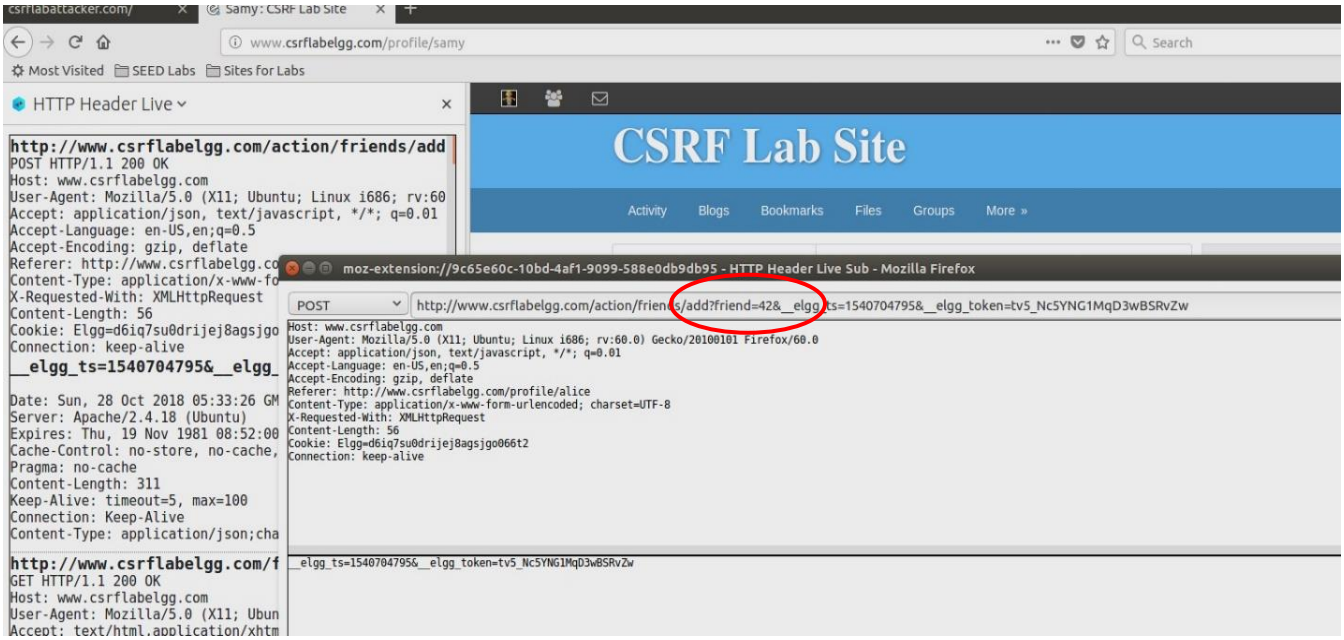

Figure 2. Identifying Alice's GUID

From Figure 2, Boby can figure out Alice's GUID that passed as parameter (which is 42). Since Boby added Alice on his friend list. He can now send messages to Alice. At the beginning, Alice doesn't have anyone listed as friend. Boby's purpose of this CSRF attack is to become mutual friends with Alice. Therefore, Boby sends a phishing message to Alice. This message contains a HTTP request that is hidden within html tag. For this example, Boby uses img tag which is used for displaying image on web browser. Img tag consists of 2 parts, property of image (such as height and width) and source of the image. If source of the image tag is web link, browser tries to open the link to obtain the source img file. Using

this, Boby can implement Elgg web application's add friend request with Boby's GUID in img tag. This img tag needs Boby's own GUID which can be identified from browser inspection or page source that can be found in developer tool (which is 43).

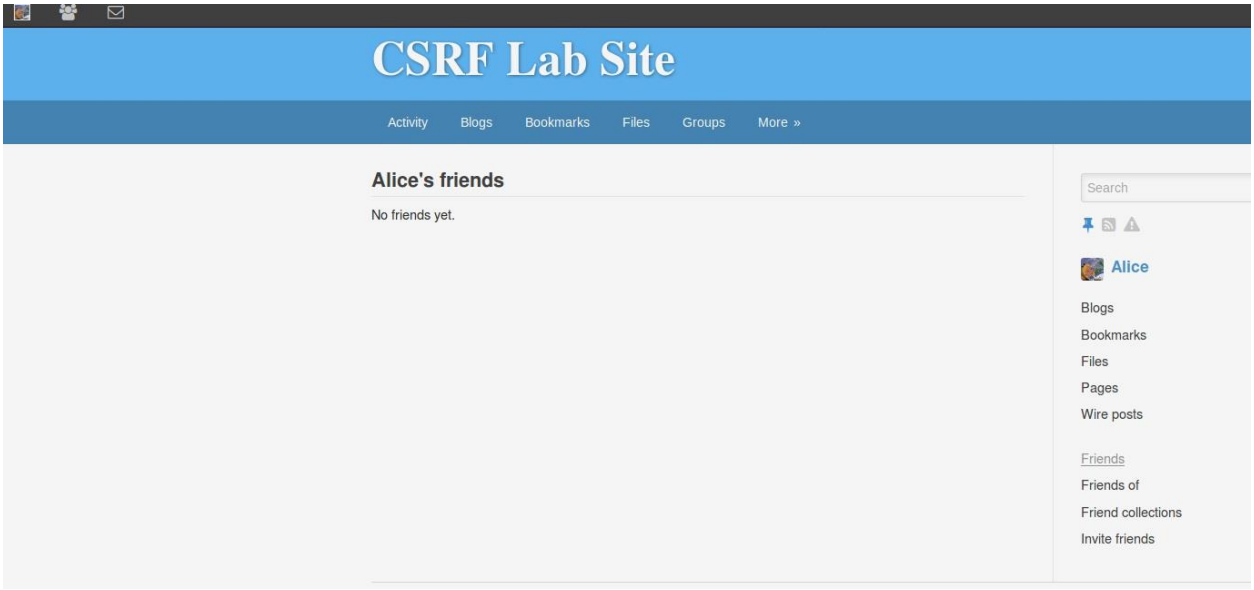

Figure 3. Alice's empty friend list before attack.

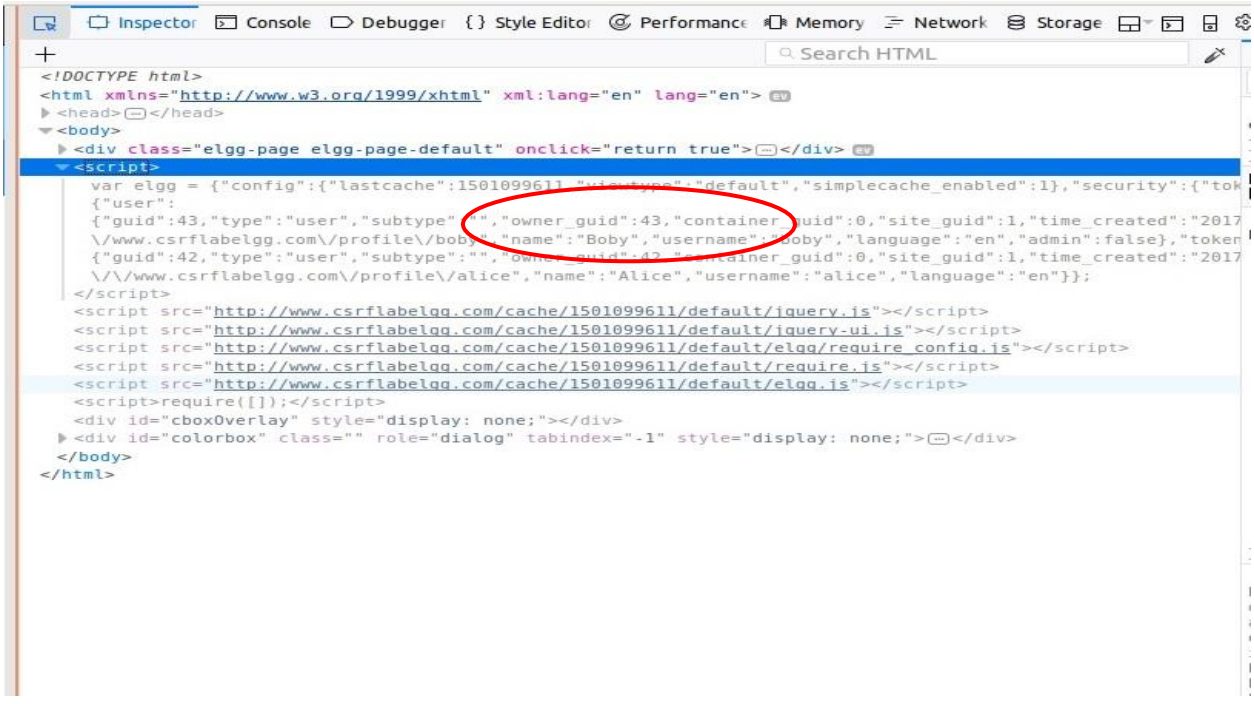

Figure 4. Boby's GUID

Boby's attack scenario is following, 1) Boby sends a malicious message with modified img tag. This tag requests the server to add user with GUID 43 as friend. 2) Alice logs in onto her account and finds a message and clicked the link. 3) Link redirects Alice to Malicious website and the img tag of the website adds Boby into Alice's friend list.

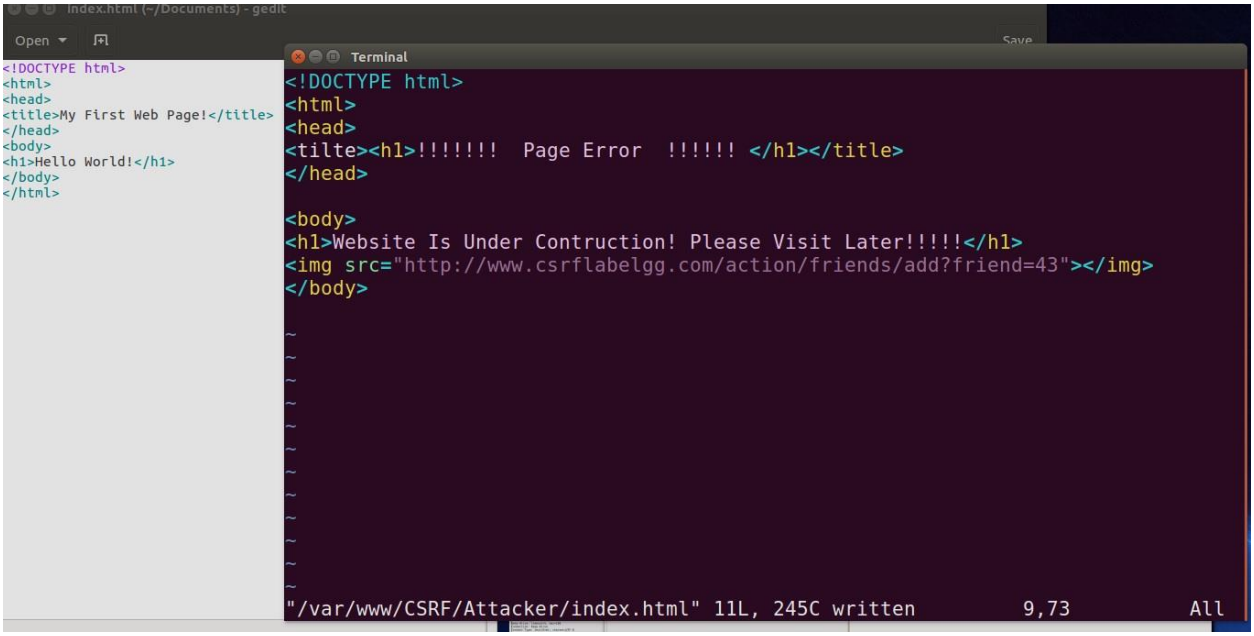

Figure 5. Boby constructs a website containing malicious img tag. The tag adds the user with GUID 43 into account owner's friend list.

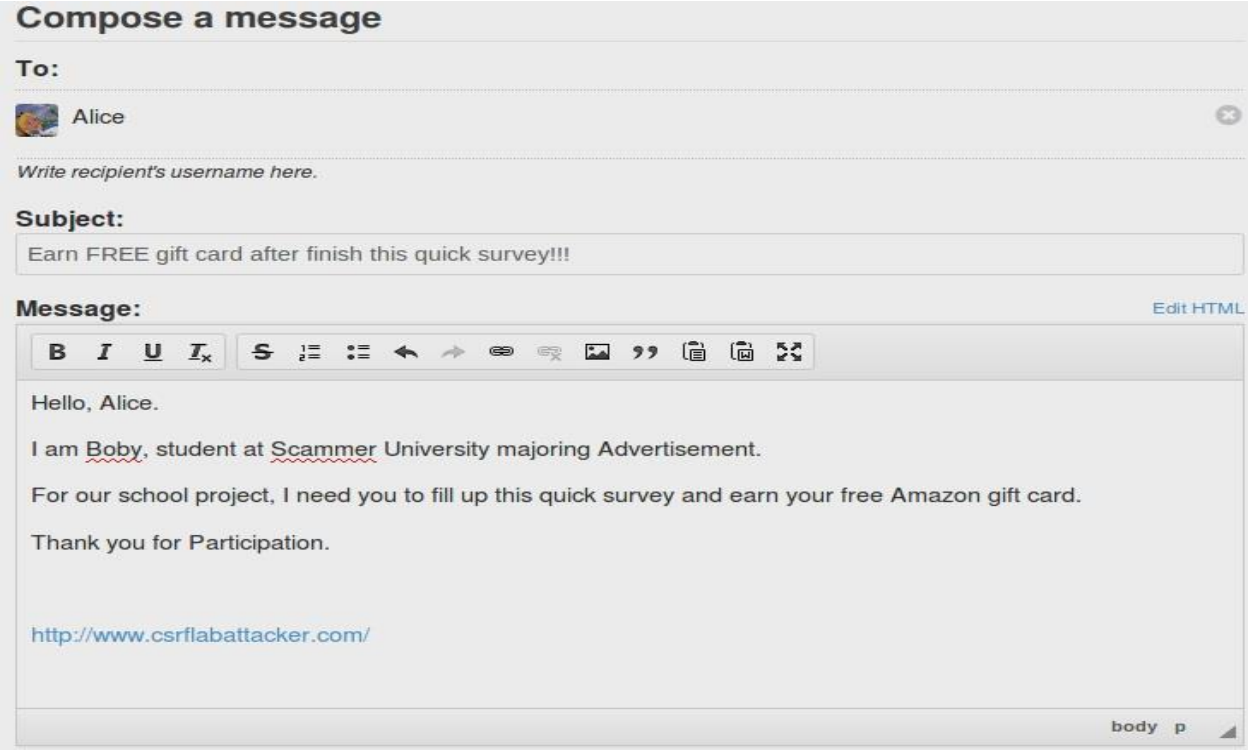

Figure 6. Boby sends a message with malicious website for CSRF attack.

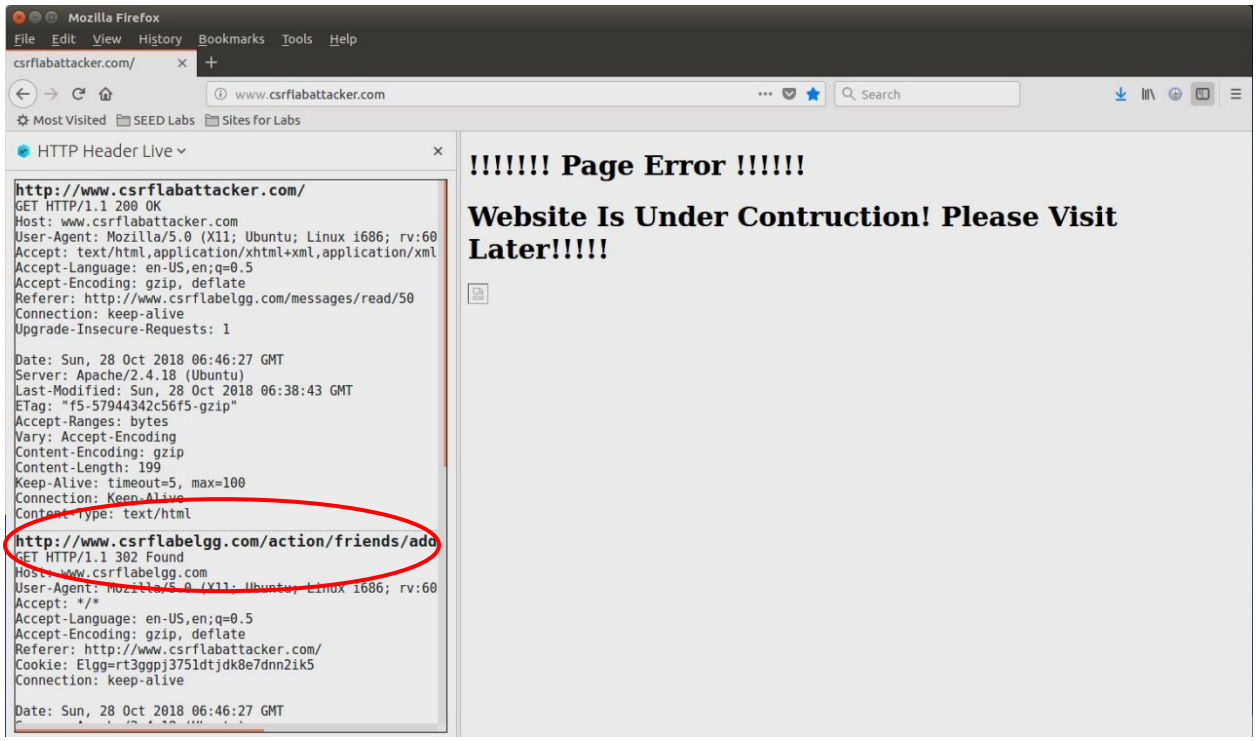

Figure 7. Alice clicks link and redirected to malicious website. Img tag shows error since the source of this tag is not an img file. The malicious website send friend add request to Elgg website.

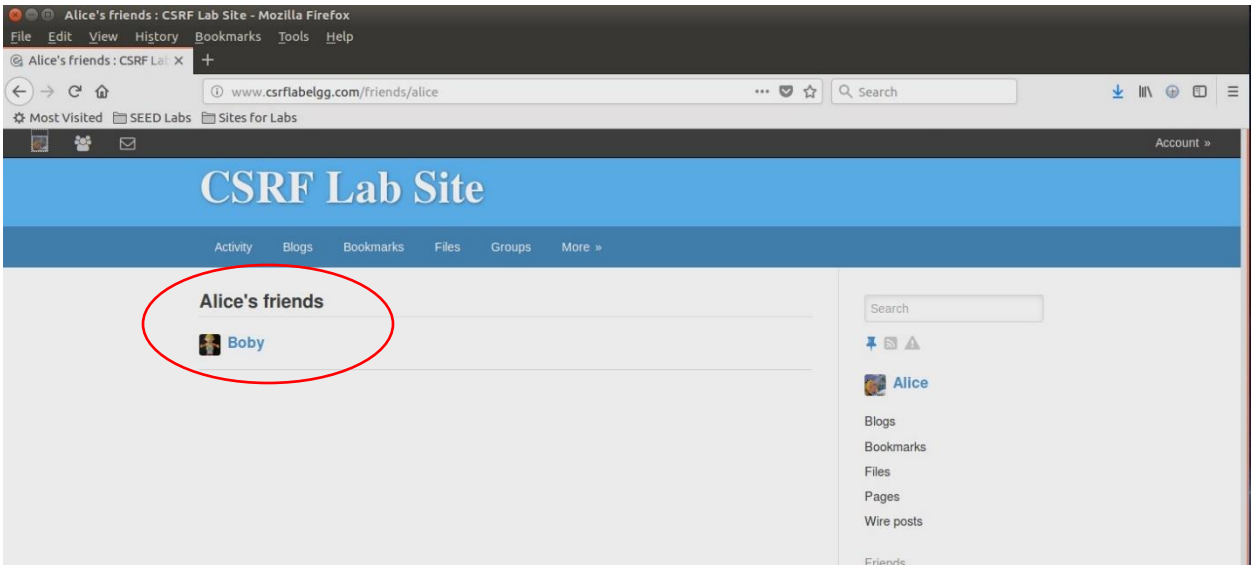

Figure 8. As a result, Alice now friend with Boby.

Countermeasures

There are two well-known methods for counter CSRF attack. First one is using Secret Token and second one with Referrer Header. Elgg website uses secrete token that is generated with session id, user id and time stamps. This token verifies a request to be a genuine request made from Elgg site not from other sites.

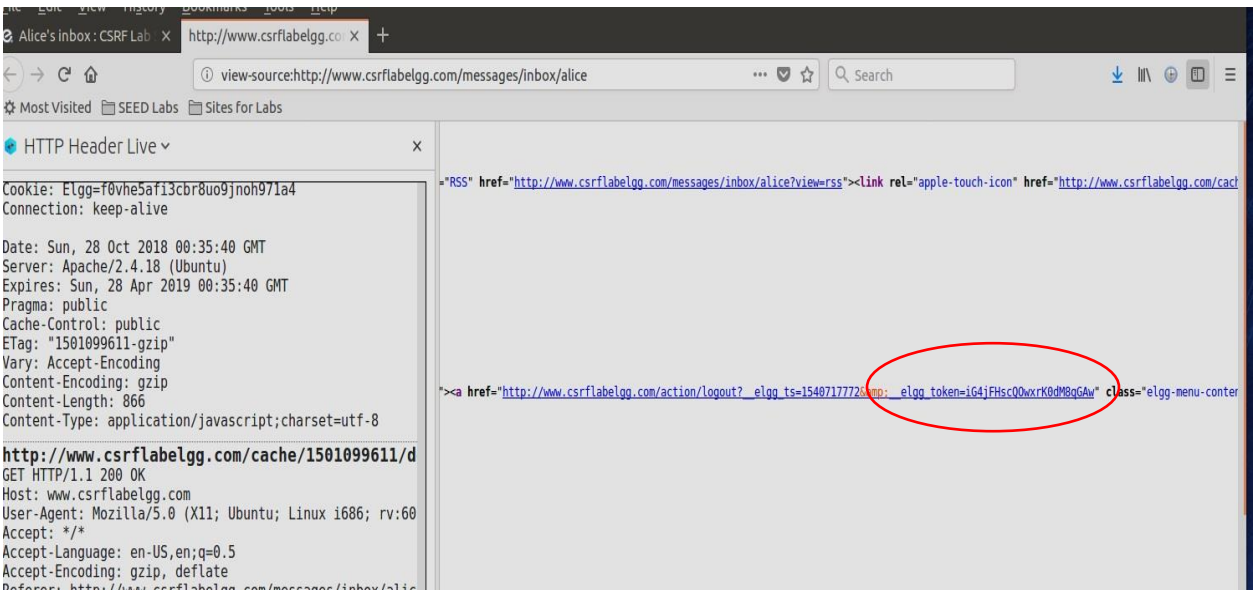

Figure 9. Elgg token is successfully created.

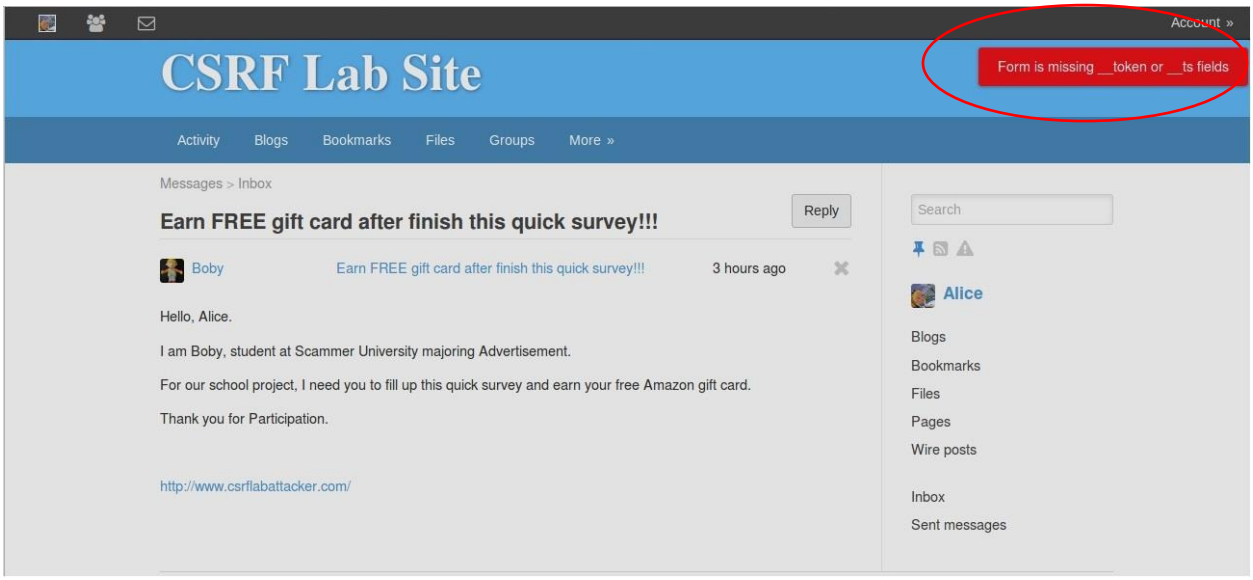

Figure 10. Missing token successfully block CSRF attack.

Conclusion

After 8 hours of devotion to this project, I acquired basic principle of CSRF attack and how a client protects its users from CSRF. Sometimes, it was hard to find desired traffic using HTTP Header Live app since lack of search option. In this case using developer's tool is better. Using linux's vi editor was the hardest part of this because of lack of background knowledge about linux.