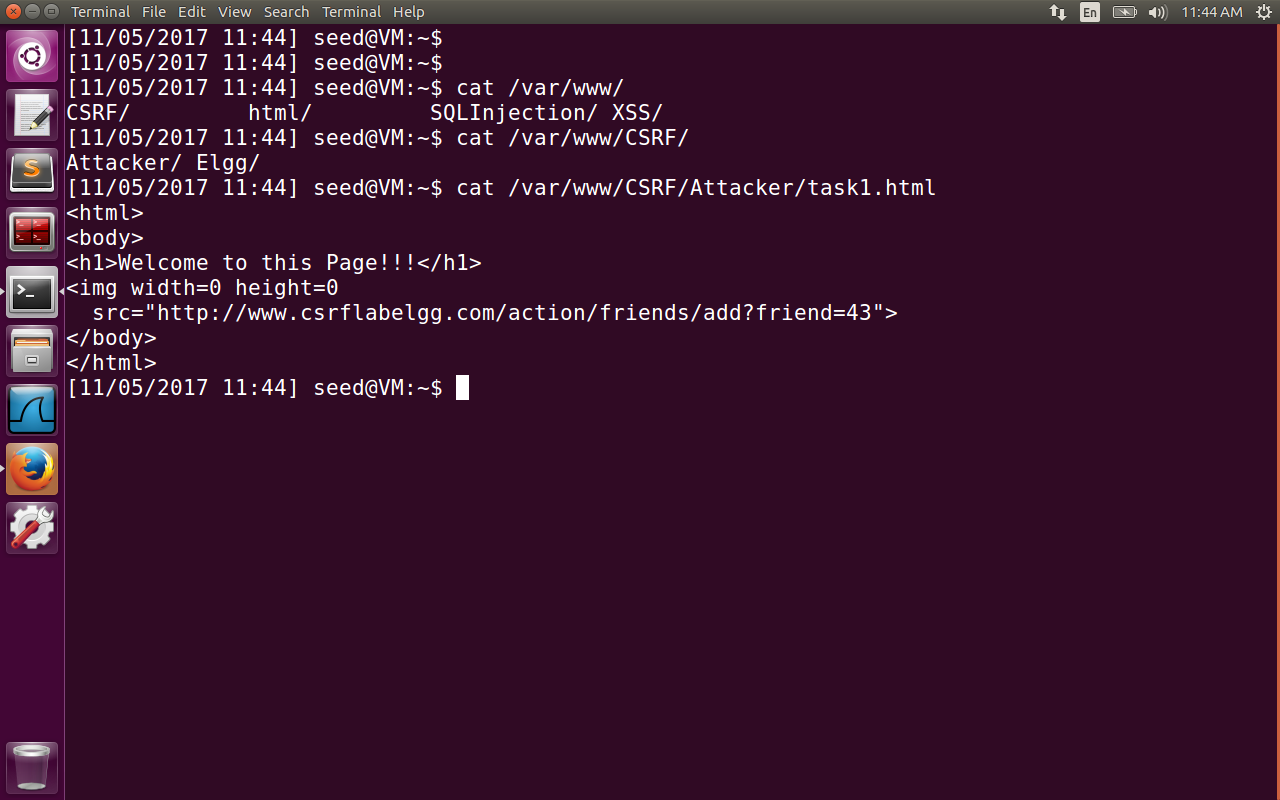
CROSS-SITE REQUEST FORGERY (CSRF) Attack Lab

(Web Application: Elgg)

NAME: JASHWANTH REDDY GANGULA

SUID: 646254141

**Task 1: CSRF Attack using GET Request**

****

First the apache web server is started as listed in the lab using the command ‘sudo service apache2 start’. Apache is configured with 2 DNS servers.

|  |  |
| --- | --- |
| <http://www.csrflabattacker.com/> | /var/www/CSRF/Attacker/ |
| <http://www.csrflabelgg.com/> | /var/www/CSRF/Elgg |

The folder /etc/apache2/sites-available/000-default.conf is configured with Each VirtualHost containing the server name and document root as follows.

<VirtualHost \*:80>

ServerName <http://www.csrflabelgg.com>

DocumentRoot /var/www/CSRF/Elgg

</VirtualHost>

Now we modify /etc/hosts files to map these two urls to localhost (127.0.0.1).

When Alicy logs onto the elgg social networking site, an active session is created. A cookie is created on the web browser with session identifier. Web browsers attach this cookie even if it is a cross-site request. In the first task, we will need to find the guid of Boby so that when the alice clicks on the malicious site, he is added as a friend by Alicy. We can find it using live HTTP headers, when Boby logs onto his system, and tries to edit his profile, the POST request contains his guid. The task1.html page contains the following code as shown in the image1.

<html>

<body>

<h1>Welcome to this Page!!!</h1>

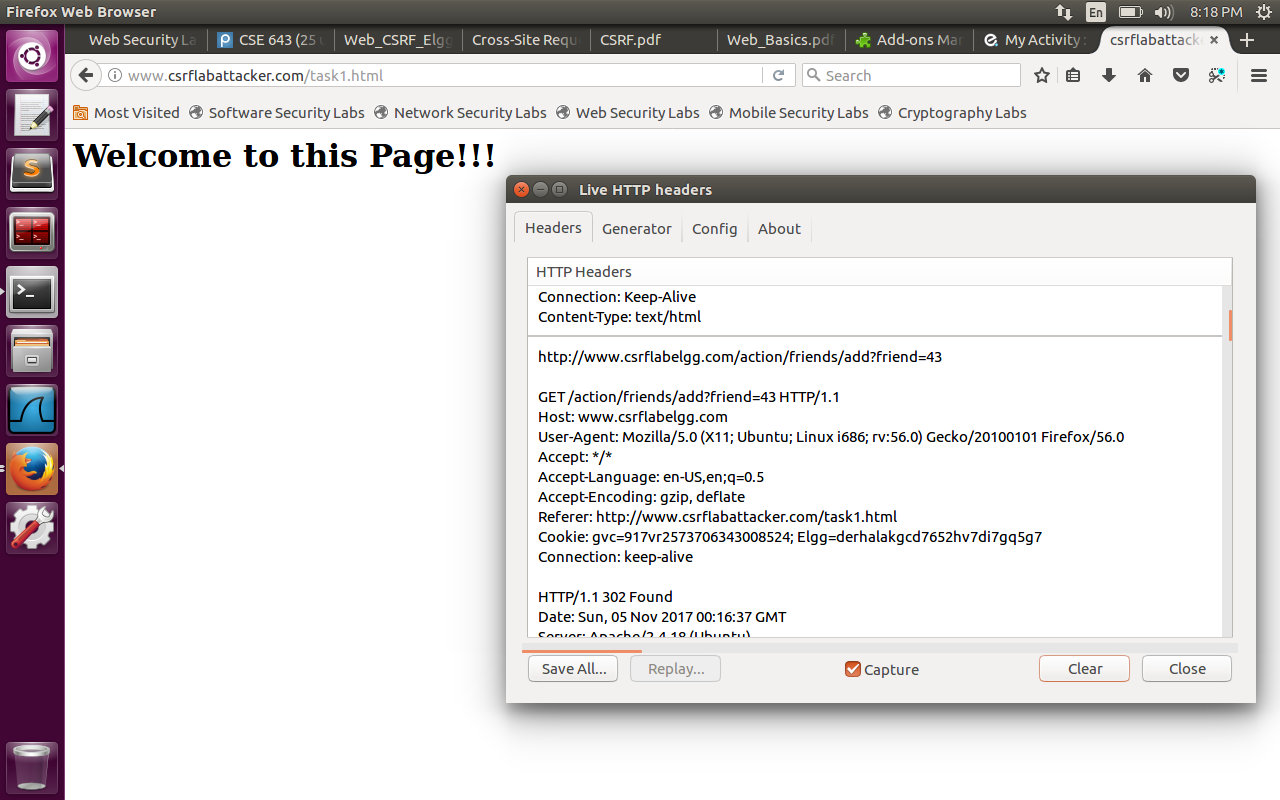
<img width=0 height=0

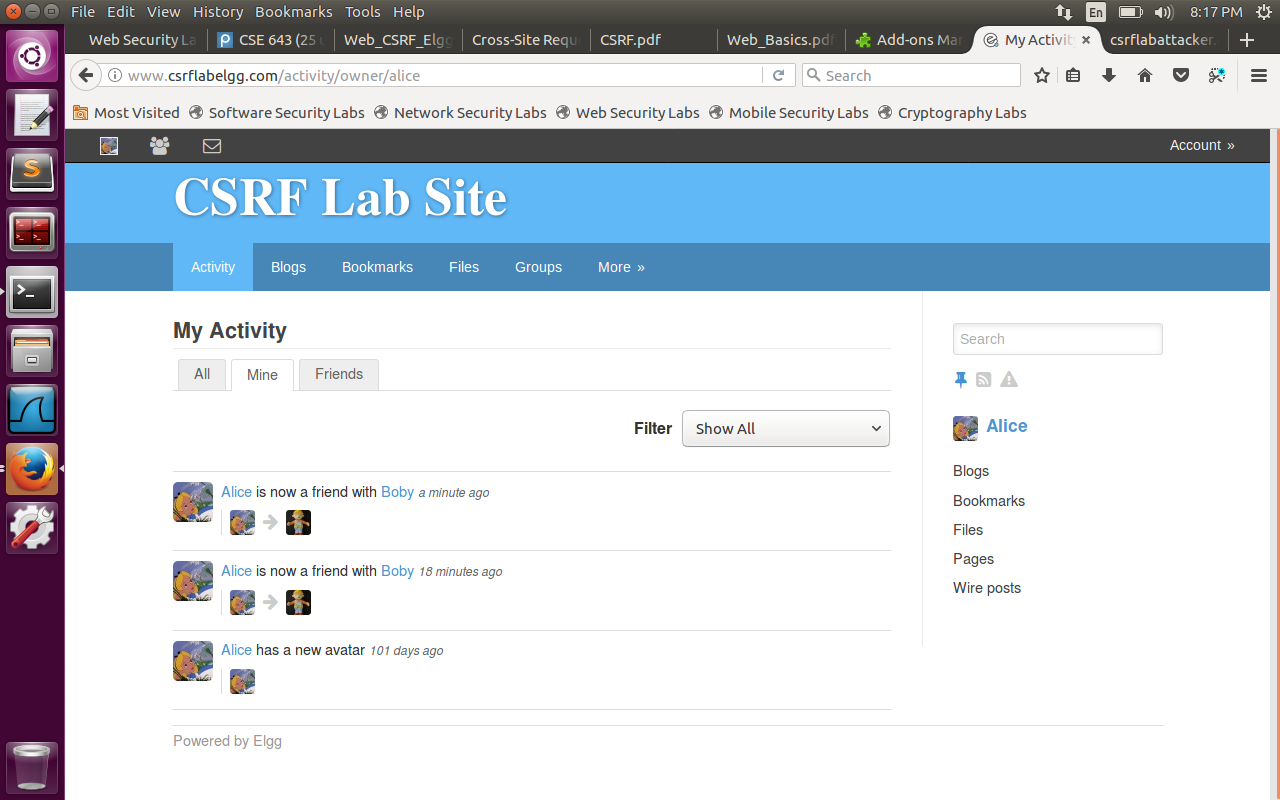
src=<http://www.csrflabelgg.com/action/friends/add?friend=43>>

</body>

</html>

When this page is sent back by the server to the browser, the browser makes a request again to fetch the image(whose width and height are zero). Here we put malicious link in the source to add the friend request as shown in the image below. The request is a get request and the reply was successful, which means Boby is now added as friend to Alicy.

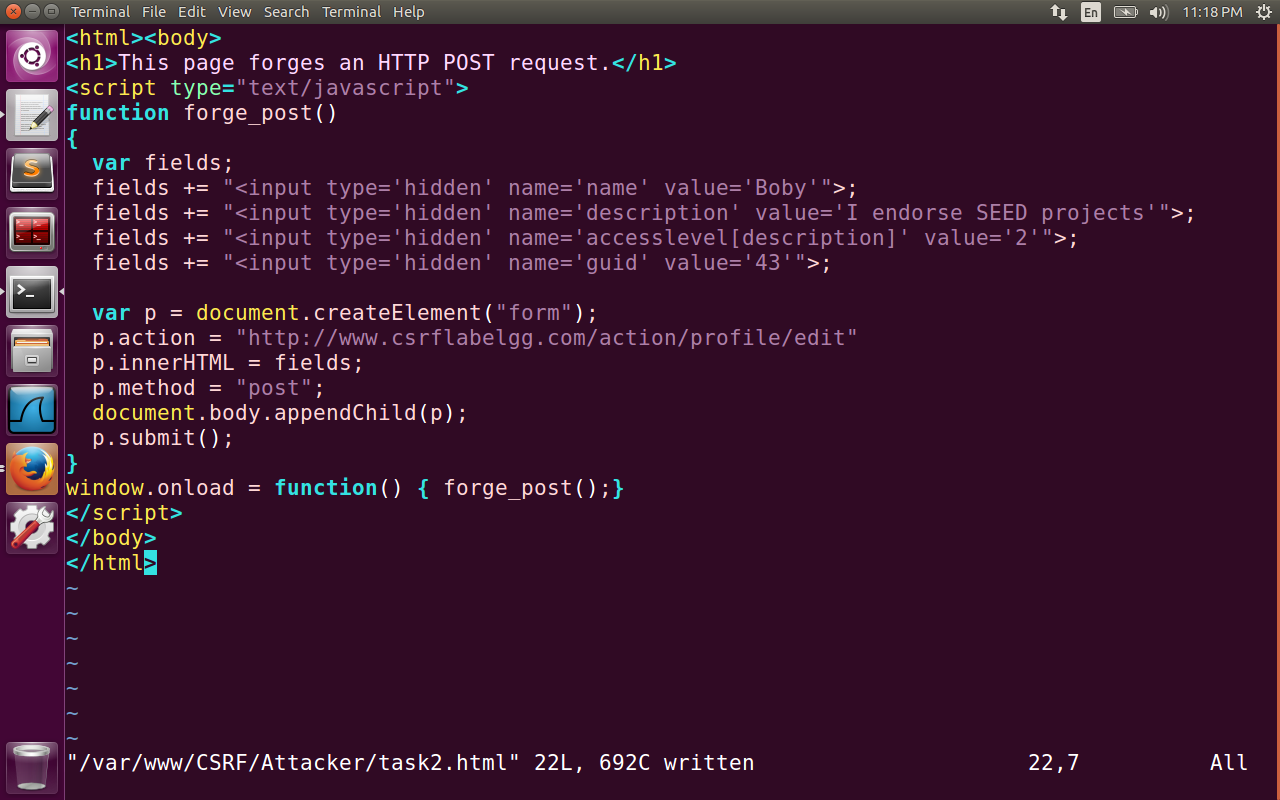
****

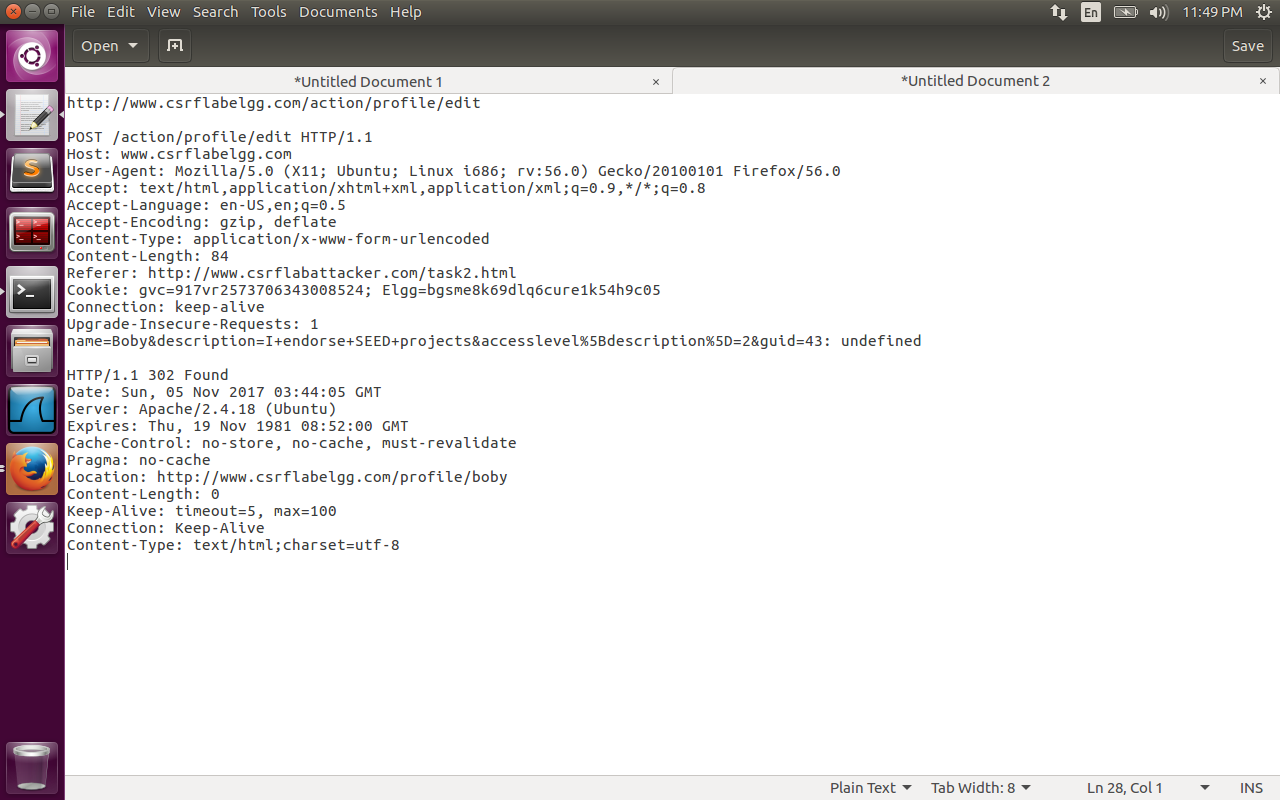
****

**4.2 Task 2: CSRF Attack using POST Request:**

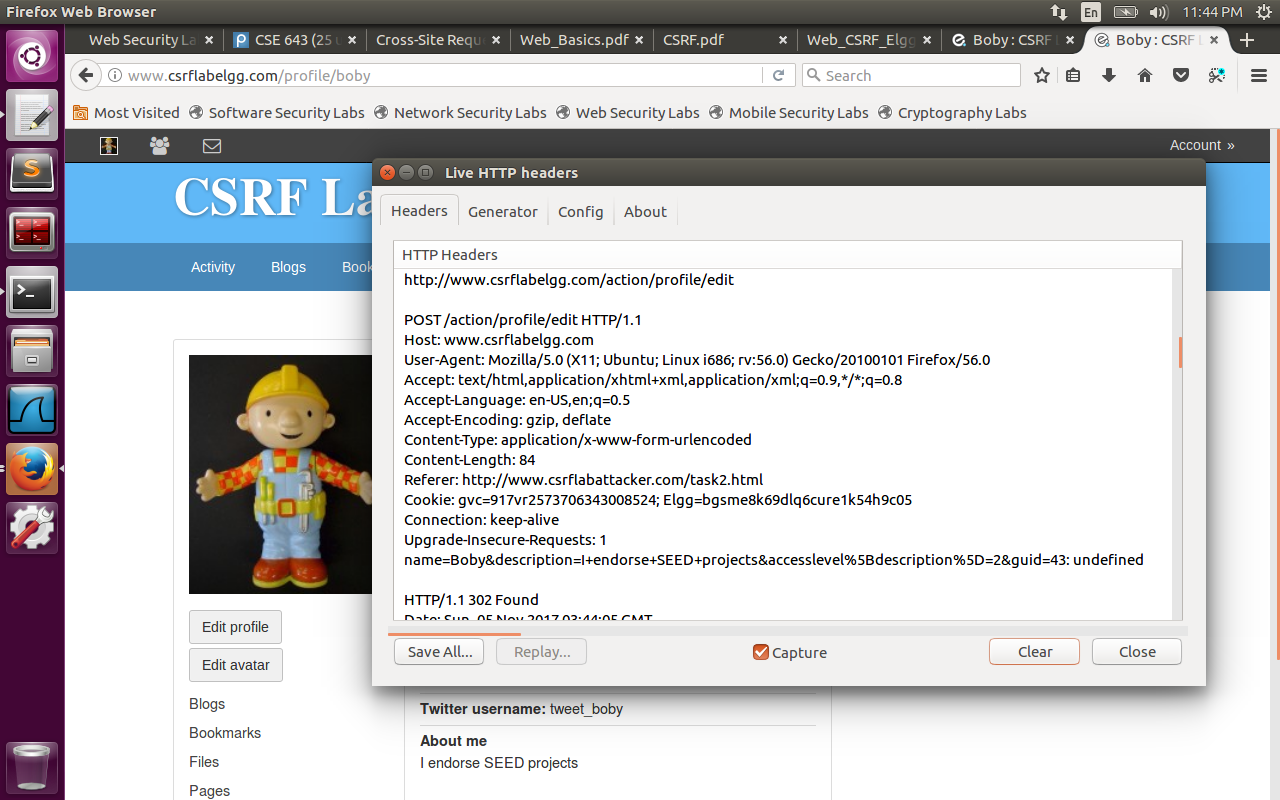
In this task, Alice attacks Boby account, after that Boby account will have ‘I support SEED projects’ on his profile account. One way is Alice sends a message which has malicious URL which constructs a post request when Boby opens the URL and his profile will be modified.

In the image shown we have created a script which we be run when the malicious page is loaded. The script tries to create a form with POST request method type. The fields in the form include the name of account (Boby), the description field should be modified to display ‘I endorse SEED projects’, the access level of description is set to 2, which implies the field is publicly available to be viewed by all. The guid of Boby is 43 as we know it from the previous task. The javascript creates a child element form inside html with all the hidden fields and sends the request to /profile/edit.php on which accepts and process the request to edit the profile.



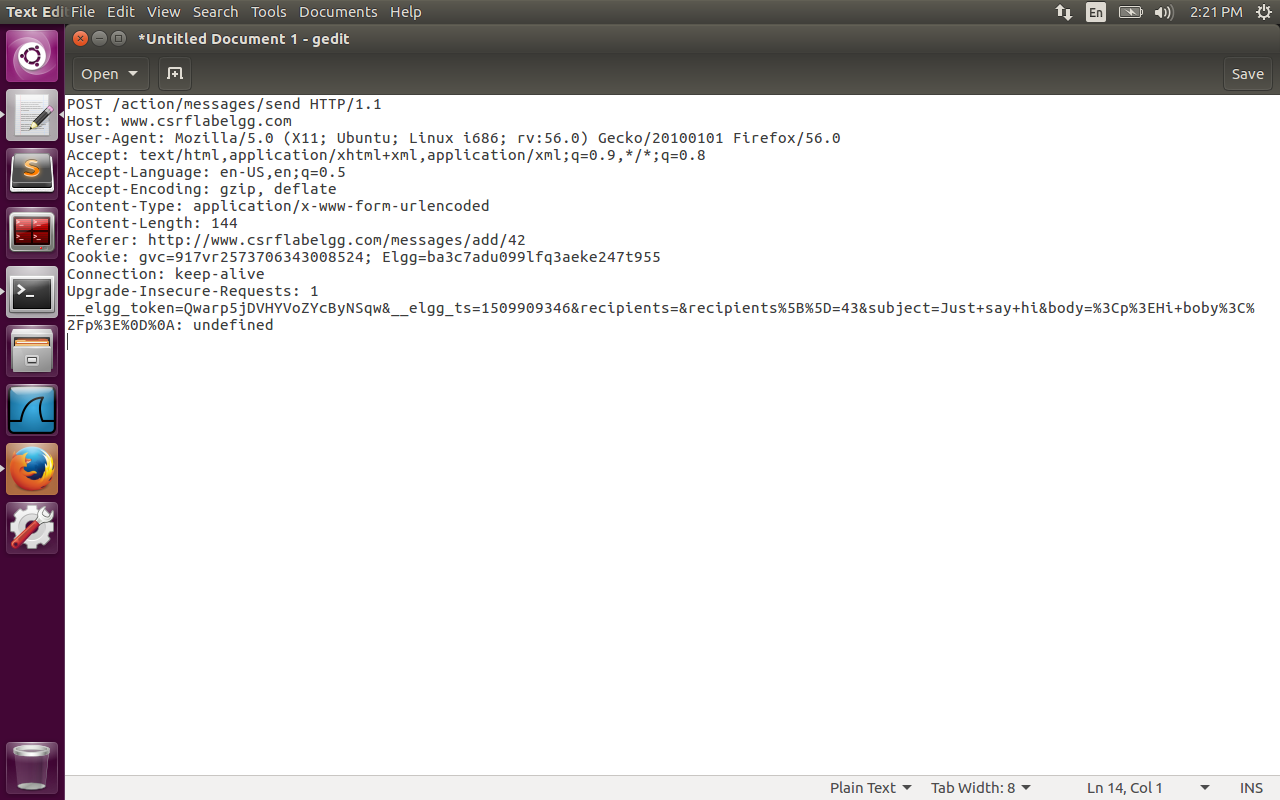


The request is shown in the image above which contains the url <http://www.csrflabelgg.com/action/profile/edit> with the method POST contains the body with form details, referrer header as <http://www.csrfattacker.com/task2.html> . The request was successful with the Boby profile damaged as shown in the image below. Please observe the image last field (About me: I endorse SEED projects).



Question 1: The forged HTTP request needs Boby’s user id (guid) to work properly. If Alice targets Boby specifically, before the attack, she can find ways to get Boby’s user id. Alice does not know Boby’s Elgg password, so she cannot log into Boby’s account to get the information. Please describe how Alice can find out Boby’s user id.

Answer: When Alice clicks on the send Message to Boby, he can inspect the HTTP header to find the guid of Boby as 43, and then sends the message to the Boby. He will now modify the malicious url with id of Boby to make it successful. In the below image, Alice inspects the live HTTP header to find the recipient id as 43.



Question 2: If Alice would like to launch the attack to anybody who visits her malicious web page. In this case, she does not know who is visiting the web page before hand. Can she still launch the CSRF attack to modify the victim’s Elgg profile? Please explain.

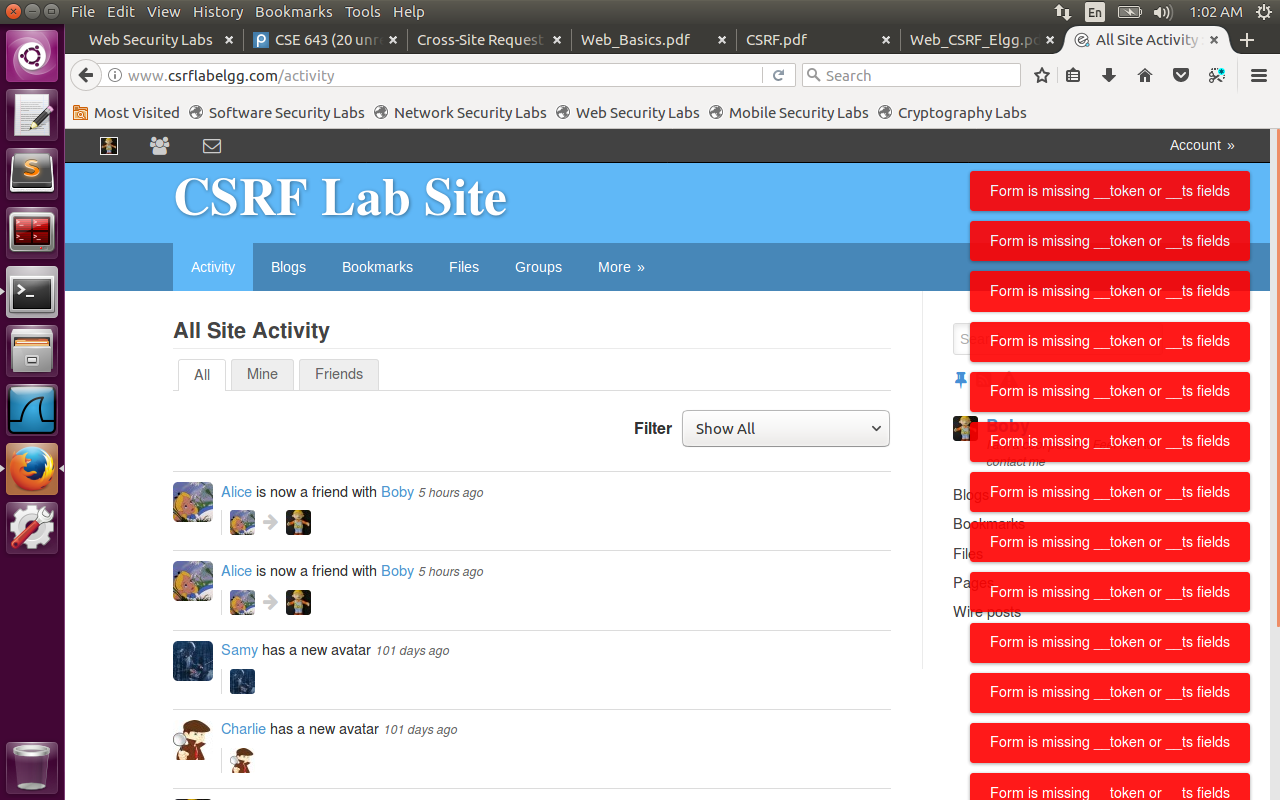
Answer: It is not possible. If somebody visits the Alice page, for Alice to attack him/her she needs the unique user id of the person. To get this information on the go and modify the malicious url to match the random user is not possible.

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

Elgg has counter measures to prevent the CSRF attack. We will use the secret token approach, where the requests coming from specific pages will contain these tokens. Cross site requests will not be able to get these tokens, and hence the attacks will be identified by the server. The two parameters  *\_\_elgg\_ts , \_\_elgg\_token* are added as body to the POST method, and appended to the GET request. These tokens are generated by views/default/input/securitytoken.php and are added to the web page. Here security token is a hash value computed using md5 message digest.

Every action from the user is validated for generated token and timestamp by the elgg application to defend against CSRF attack. If the token is present or invalid, the action is denied and the user will be redirected.

To turn on this counter measure, we go to /var/www/CSRF/Elgg/vendor/elgg/engine/classes/Elgg/ActionService.php and we comment out the ‘return true’ statement. After turning on this counter measure, we observe that the CSRF attack fails as shown in the image below.





The attacker needs the information about session id, timestamp and randomly generated string by the web application elgg which are used to create a token. The attacker cannot have all this information, as it is generated by the web site that sends the secret value to the web page during an active session. So the attacker cannot place correct tokens on the malicious url request and hence he fails.