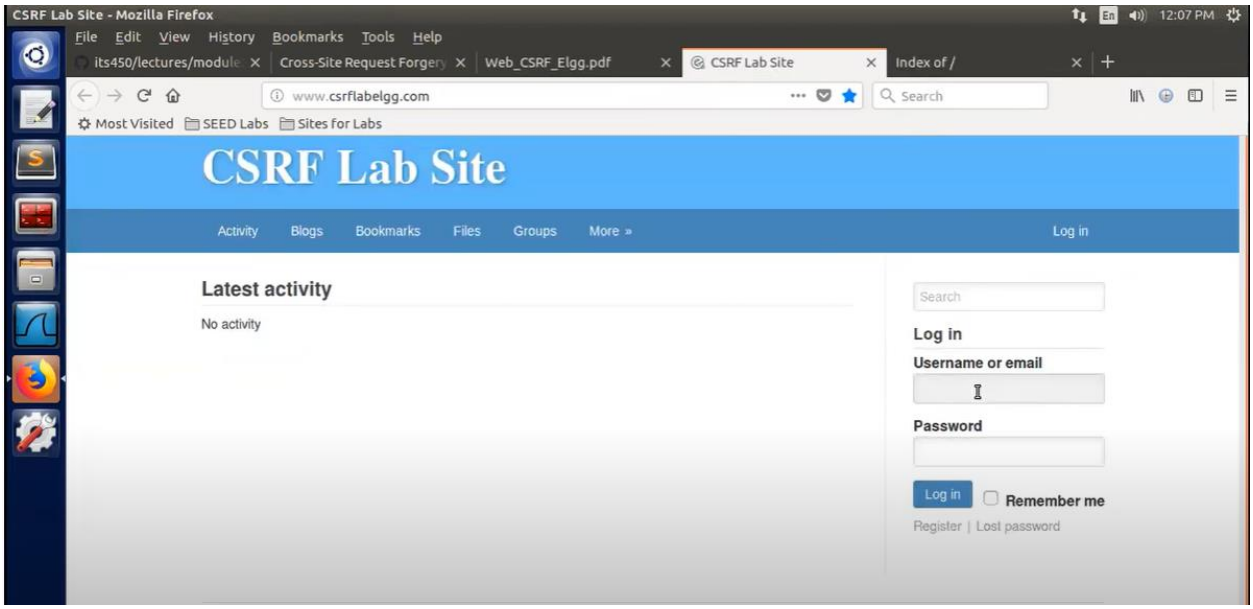
In a successful CSRF assault, the attacker causes the casualty client to carry out an activity inadvertently. For illustration, this may be to alter the mail address on their account, to alter their secret word, or to form a stores exchange. Depending on the nature of the activity, the attacker could be able to gain full control over the user's account. In the event that the compromised client contains a advantaged part inside the application, at that point the assailant may well be able to require full control of all the application's information and usefulness.

First, I have to import ubuntu seed 20.04 version in virtual box and login our scite withi given credentials.

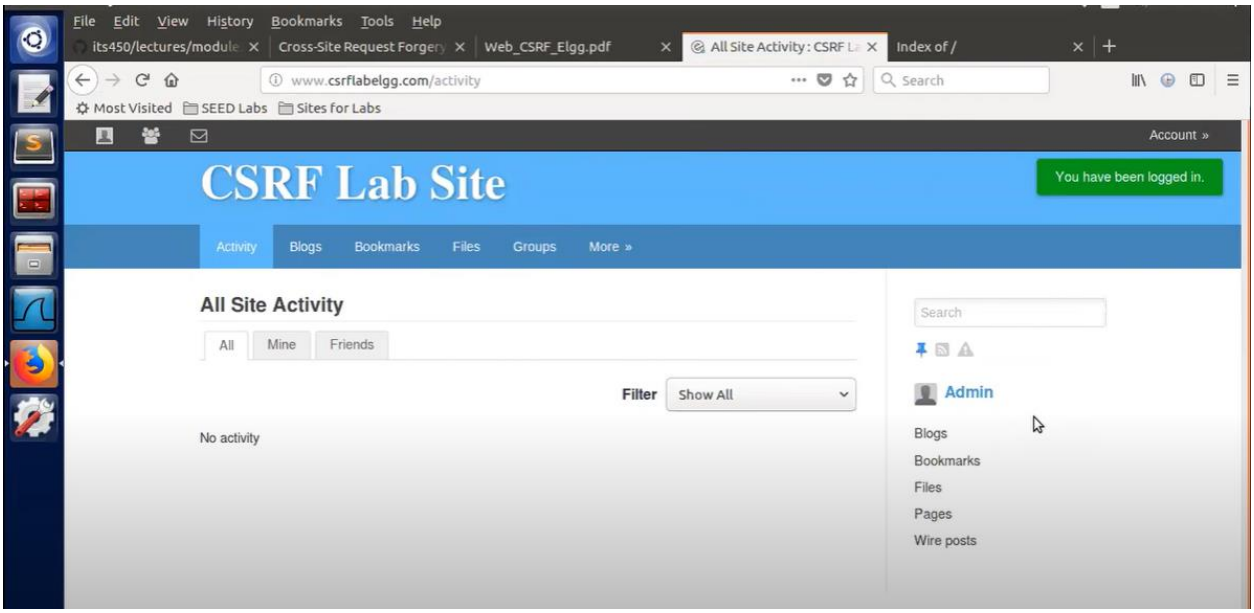


Firstly we have to use, this scite login.

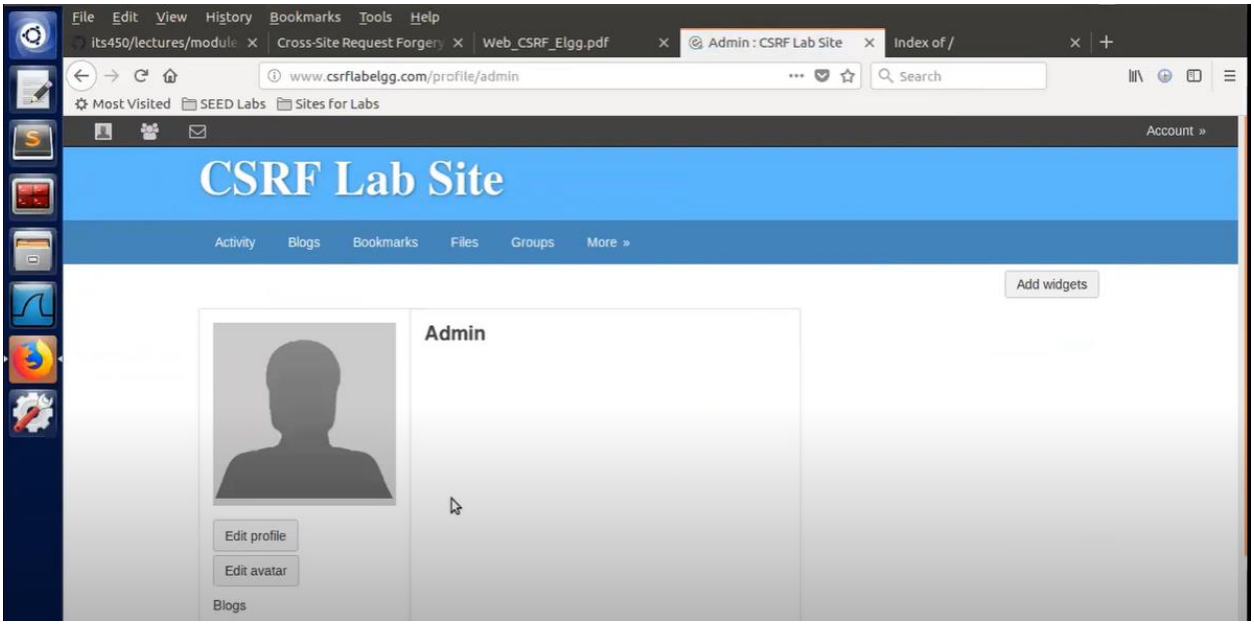
For a CSRF assault to be conceivable, three key conditions must be in place:

- A significant activity: There's an activity inside the application that the assailant incorporates a reason to initiate. This can be a advantaged activity (such as altering consents for other clients) or any activity on user-specific information (such as changing the user's possess password).
- Cookie-based session dealing with. Performing the activity includes issuing one or more HTTP requests, and the application depends exclusively on session treats to recognize the client who has made the demands. There's no other instrument in put for following sessions or approving client requests.
- No unusual ask parameters. The demands that perform the activity don't contain any parameters whose values the aggressor cannot decide or figure. For illustration, when causing a client to alter their secret word, the work isn't defenseless on the off chance that an assailant must know the esteem of the existing secret word.\

The, we will get page,



Then, we will go to profile tab,



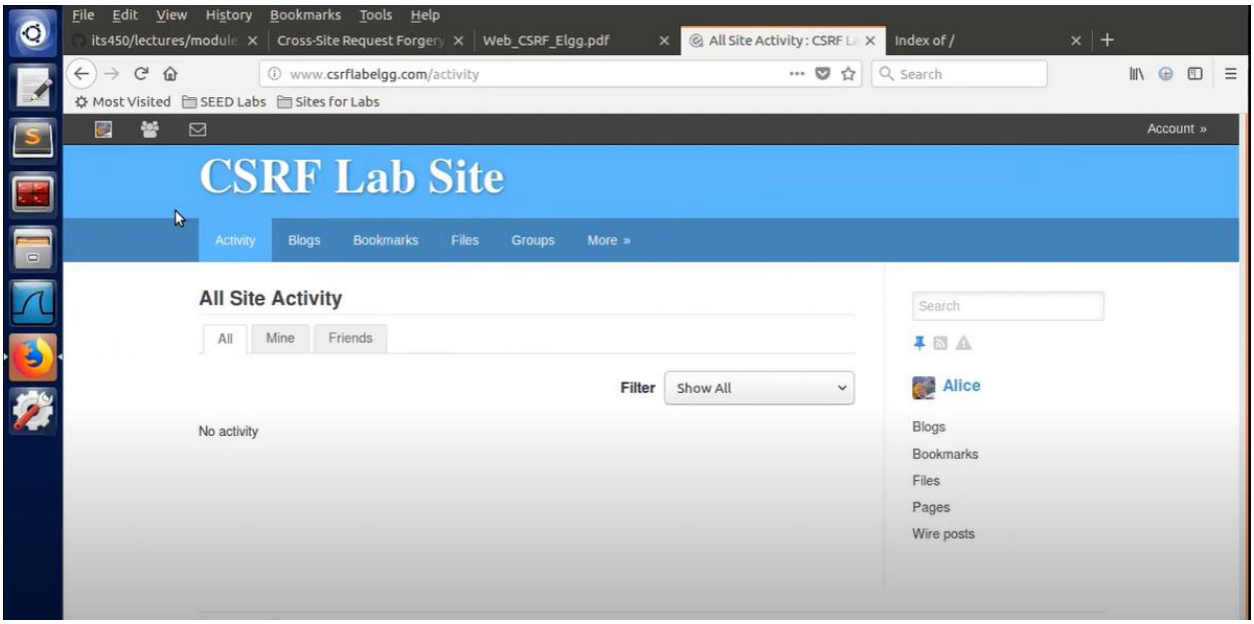
Then, we logout this account that we open using seed username. Then, we setup container using docker docker-compose.yml file.

Elgg Web Application:

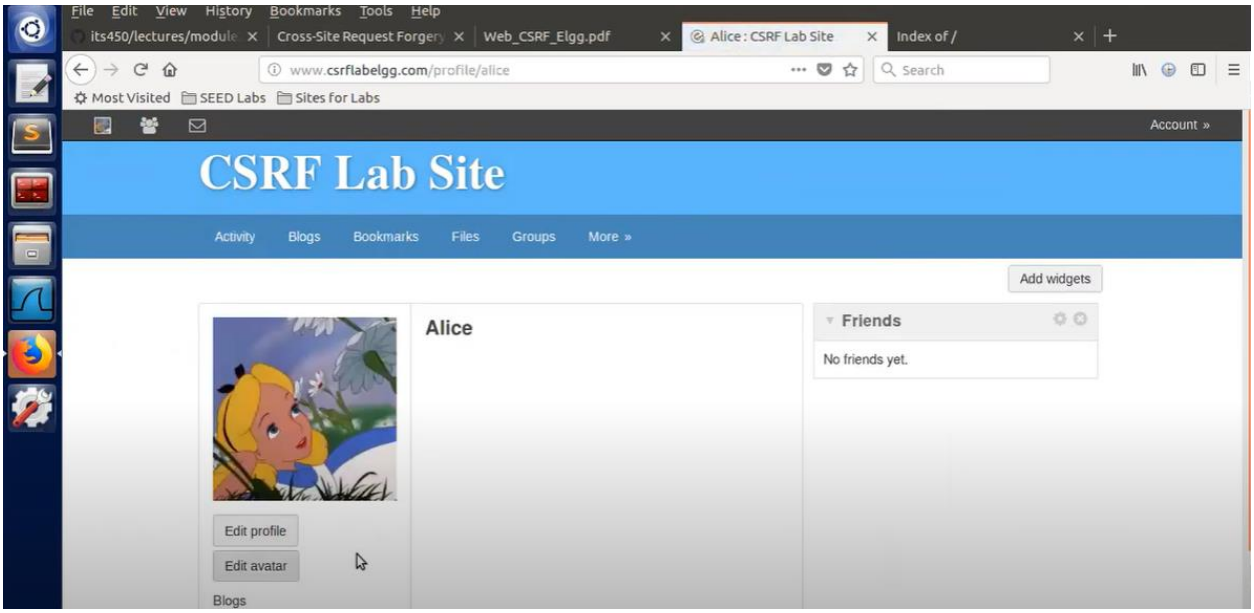
First we check the site activities: of these usernames and passwords:

```
---------------------------
UserName | Password
---------------------------
admin | seedelgg
alice | seedalice
boby | seedboby
charlie | seedcharlie
samy | seedsamy
---------------------------
```

First we check for admin then, alice



This display has been shown on screen.



This is alice profile.

We host the Elgg web application using the Apache web server. The website setup is included in

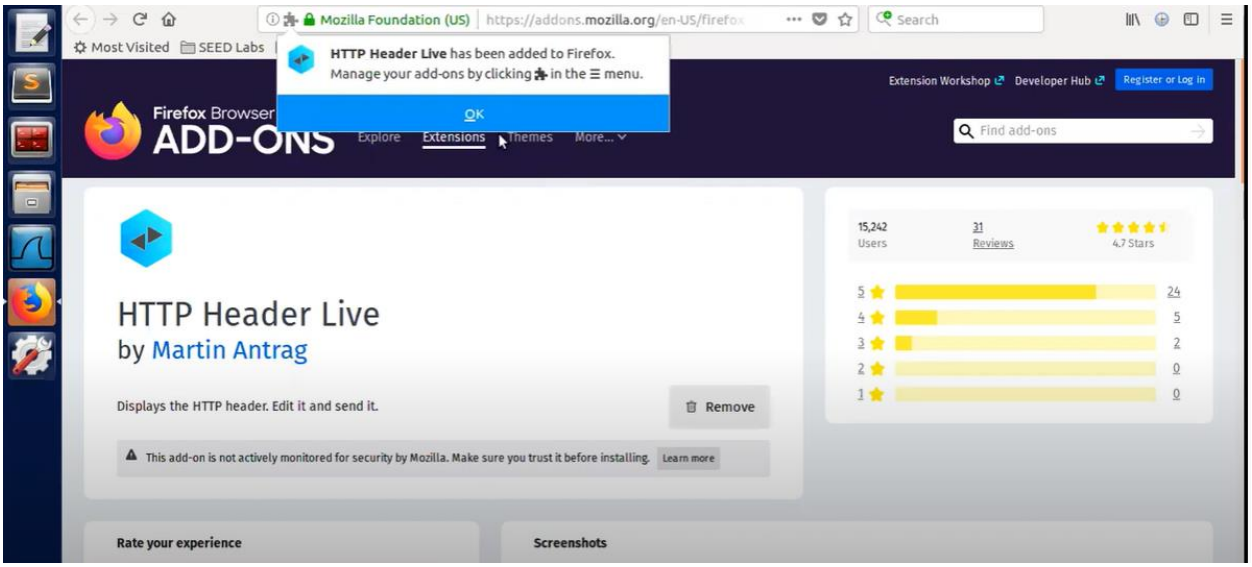
apache elgg csrf.conf inside the Elgg image folder.

This meets the conditions required for CSRF:

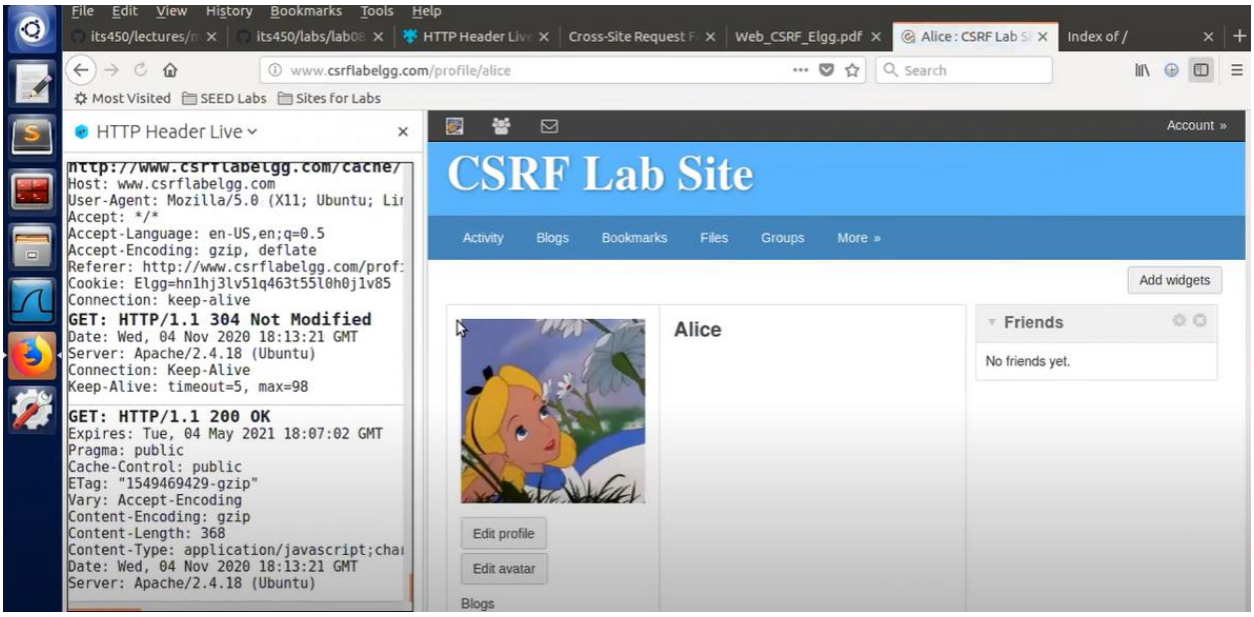
The activity of changing the mail address on a user's account is of intrigued to an assailant. Taking after this activity, the assailant will ordinarily be able to trigger a secret word reset and take full control of the user's account. The application employments a session cookie to recognize which client issued the ask. There are no other tokens or components in put to track client sessions. The aggressor can effortlessly decide the values of the ask parameters that are required to perform the activity.

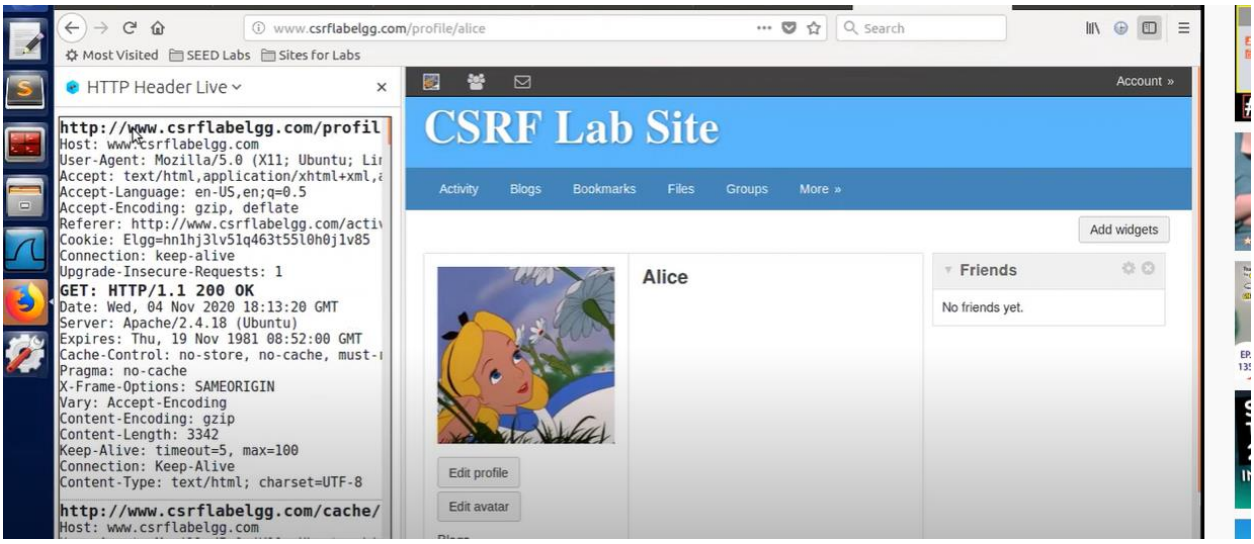
Task 1: Observing HTTP Request.

For this . First we have to add firfox to http header live server.



Then, I have opened http on alice page, after opening http header live like this shown as belo,

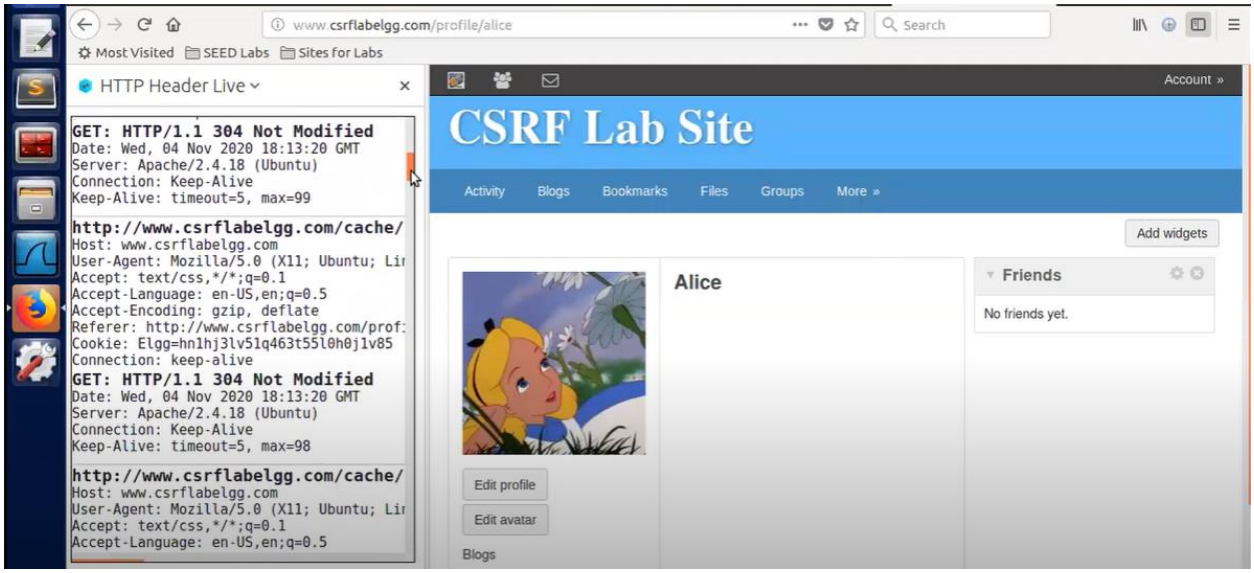


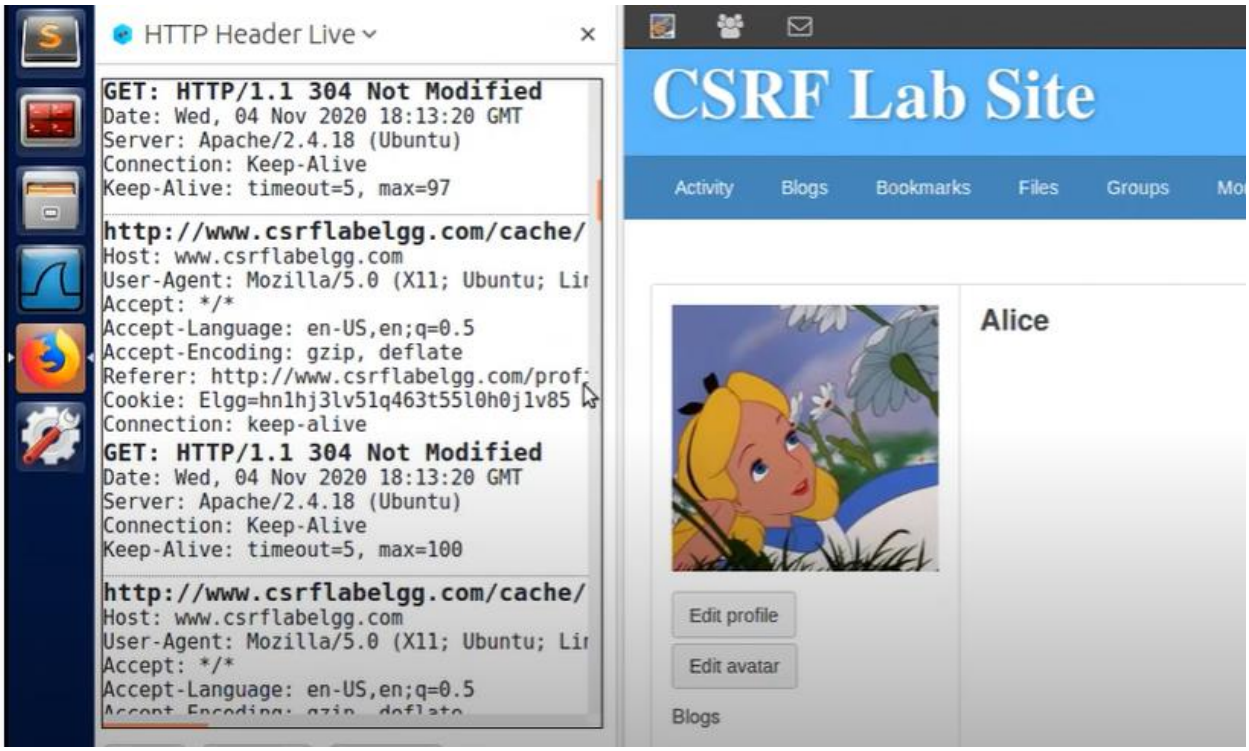


We will get the server responses on vulnerable scites while sending server request. Then refresh this page you send a get http o this server you can see from this part, with the head part, also this get command send to the server to enable the database replied with the status okay.

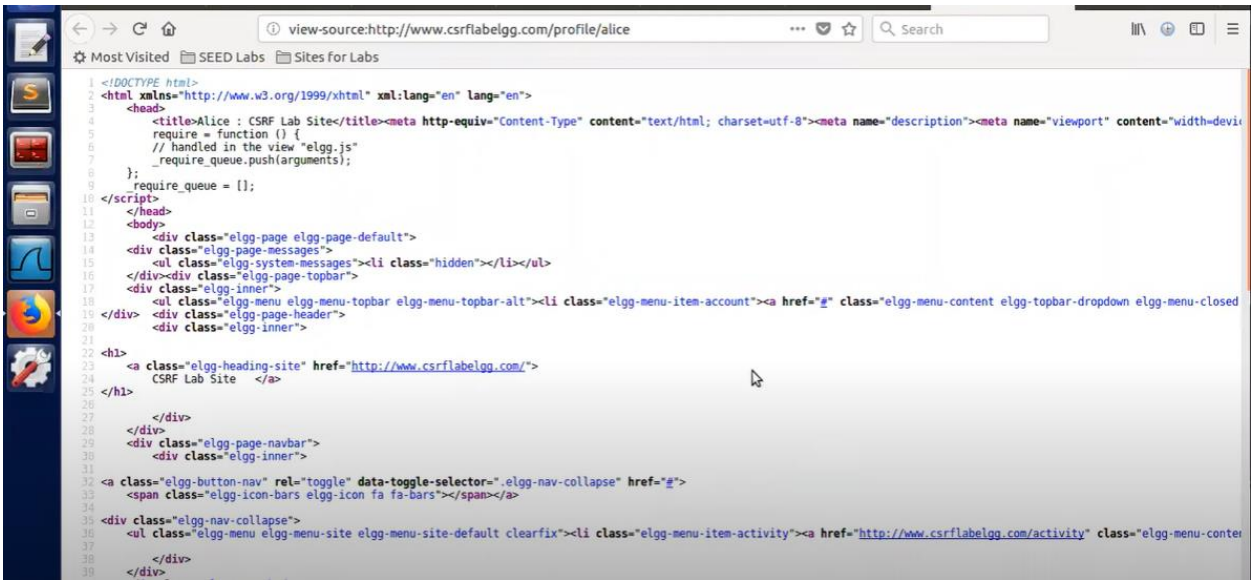
Right for this alice profile, and we see some others cache some invisible stuff.

When scroll down , cache is shown,

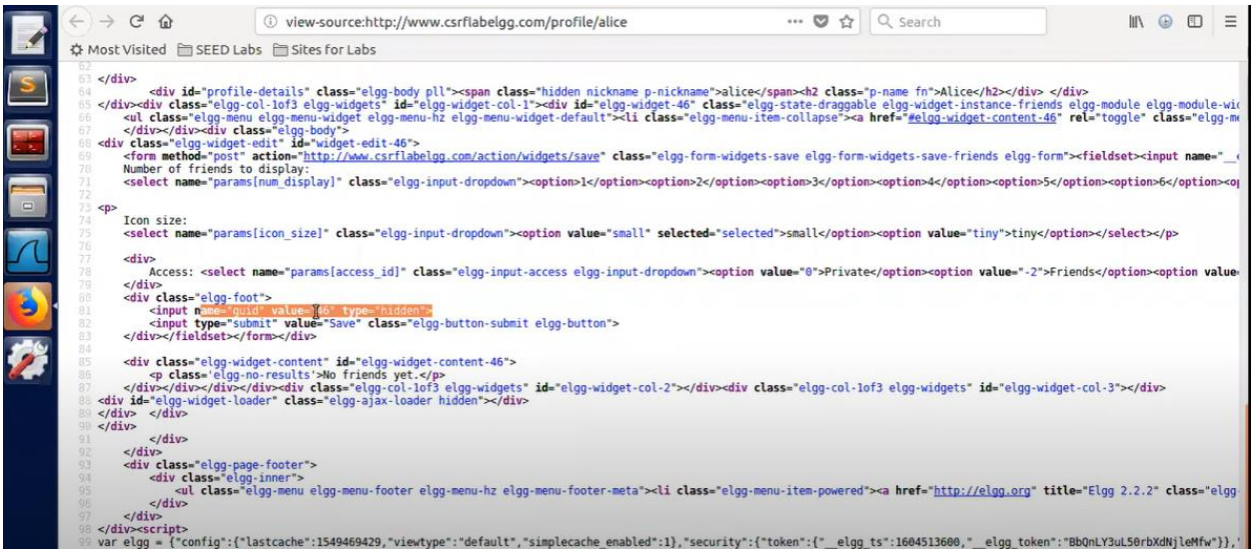




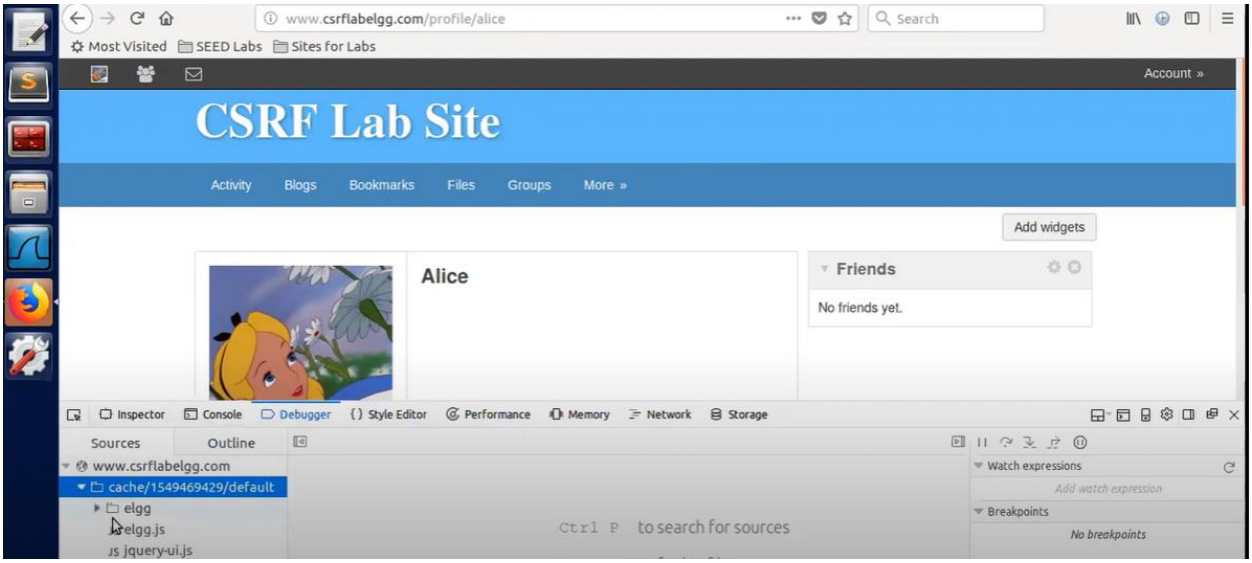
Then I have inspected the page to see the source code,

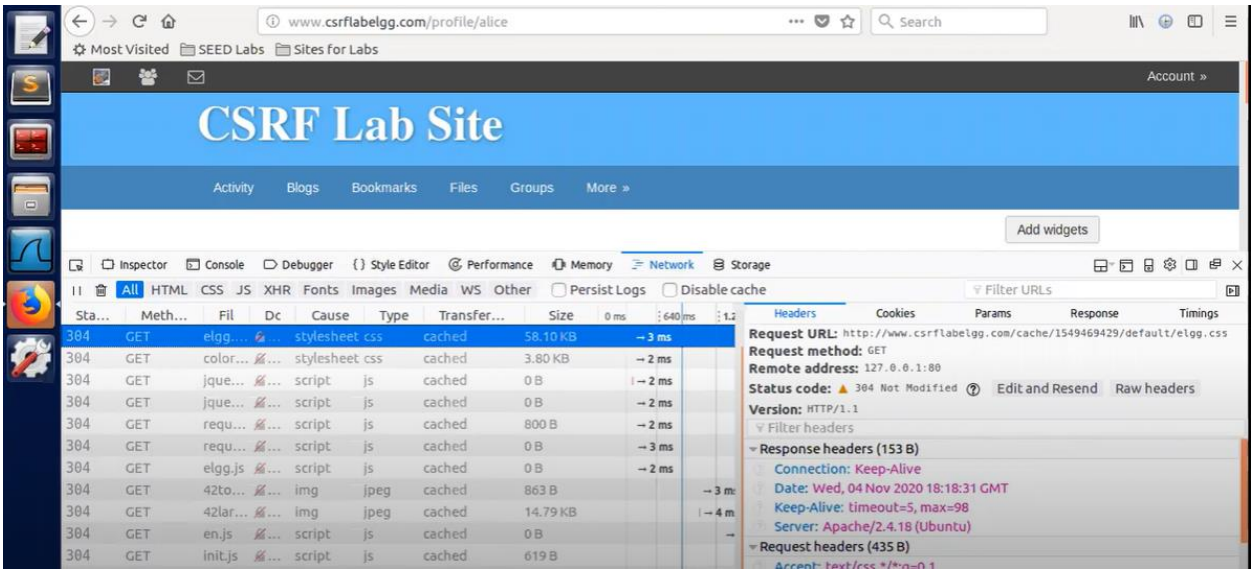


You see this script we have our eltg available compliance information with the last cache number will type security token, in the direction these are kind od measures fight against csrf right we have a time stamp for token, so these are the current measures.

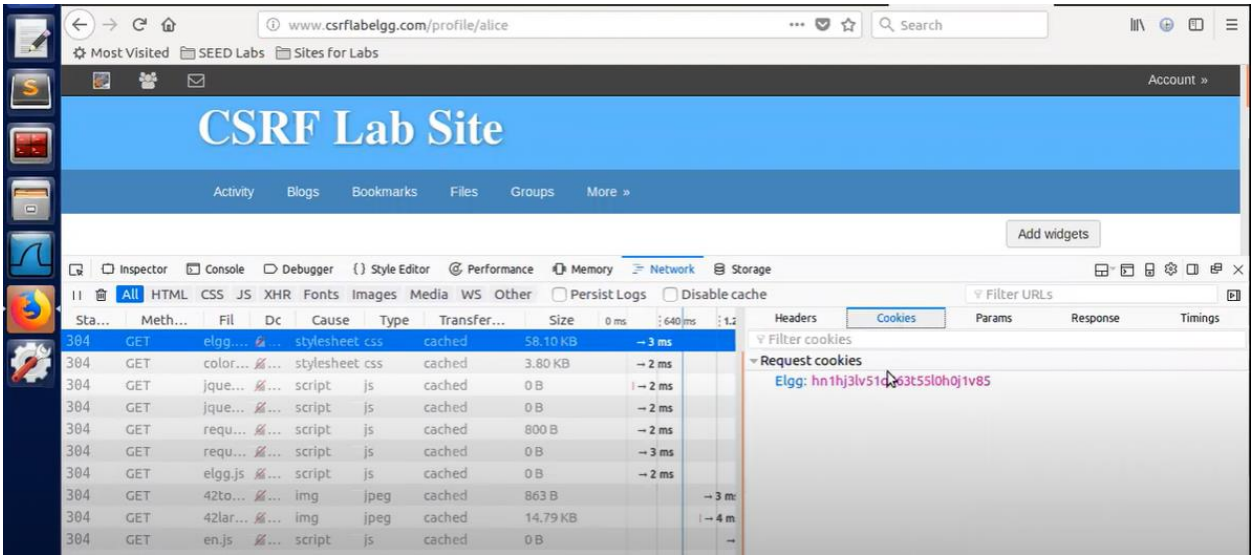


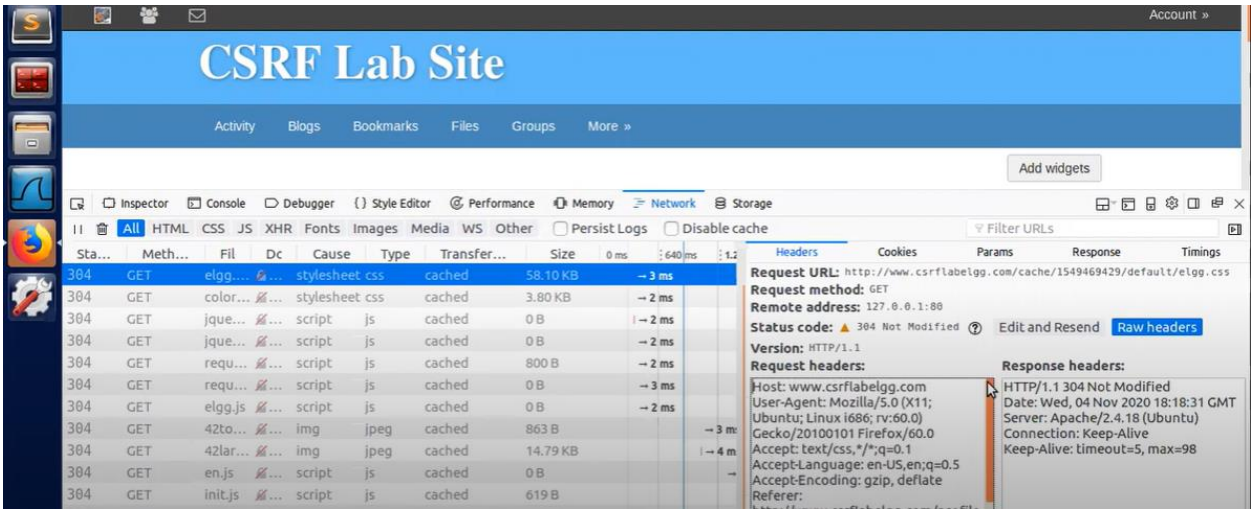
Now we have to check its cache id and javascript code. Then, refresh the code from network button to start performance analysis of every user.





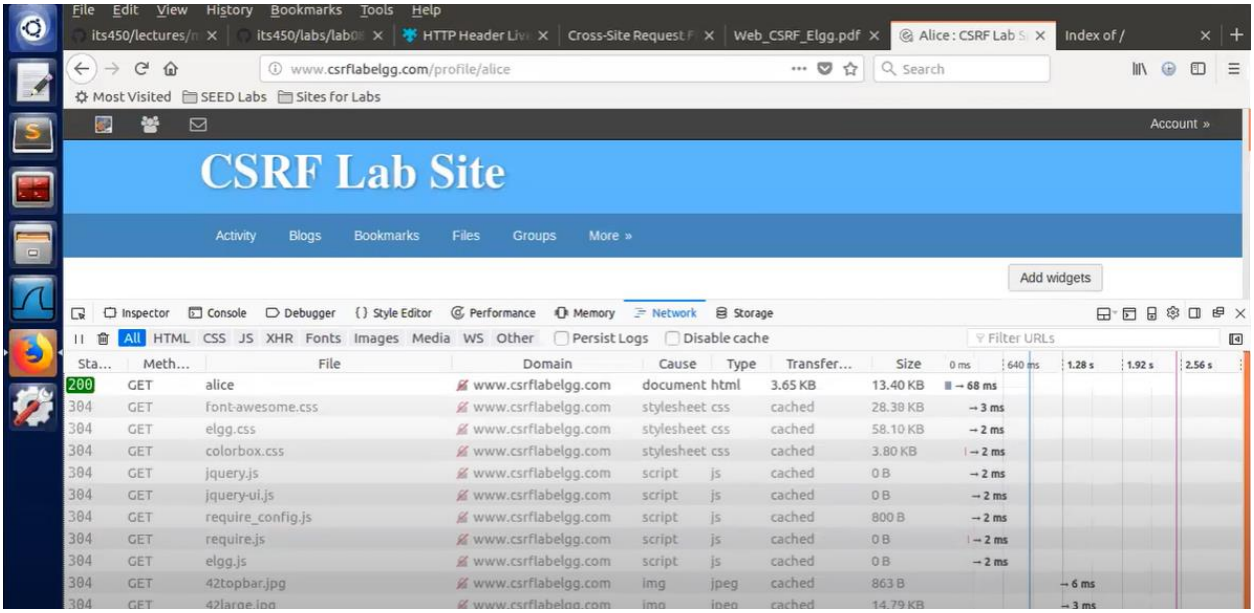
So, the response header is sent from server to our browser and then request header is sent from our browser to the server. Here you see the cookie.

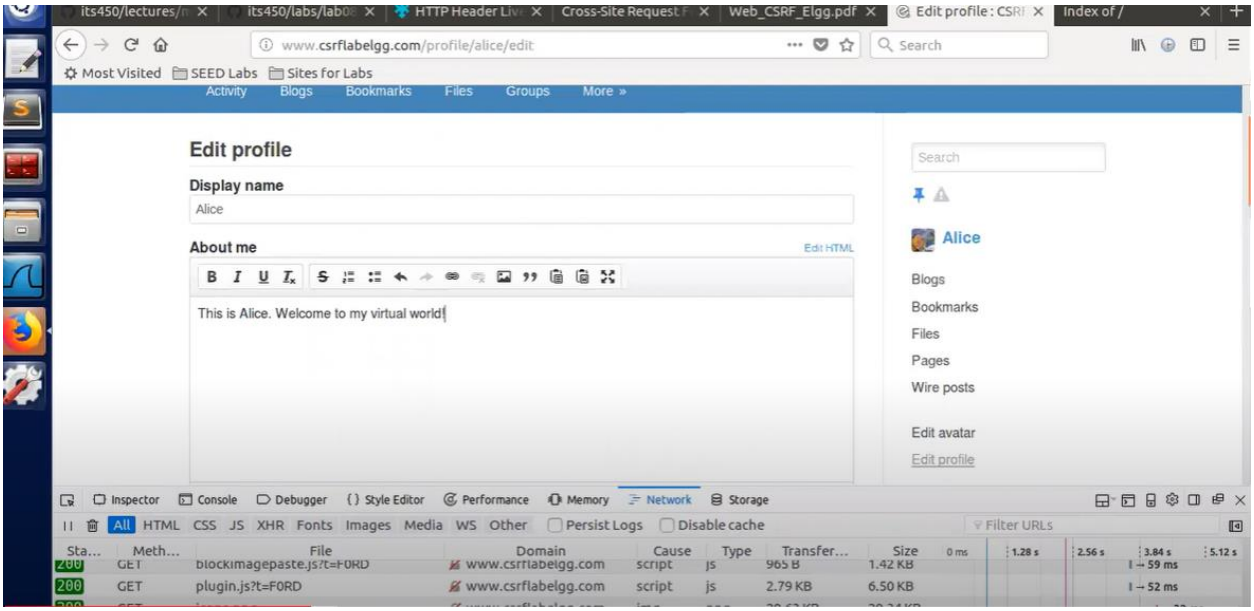


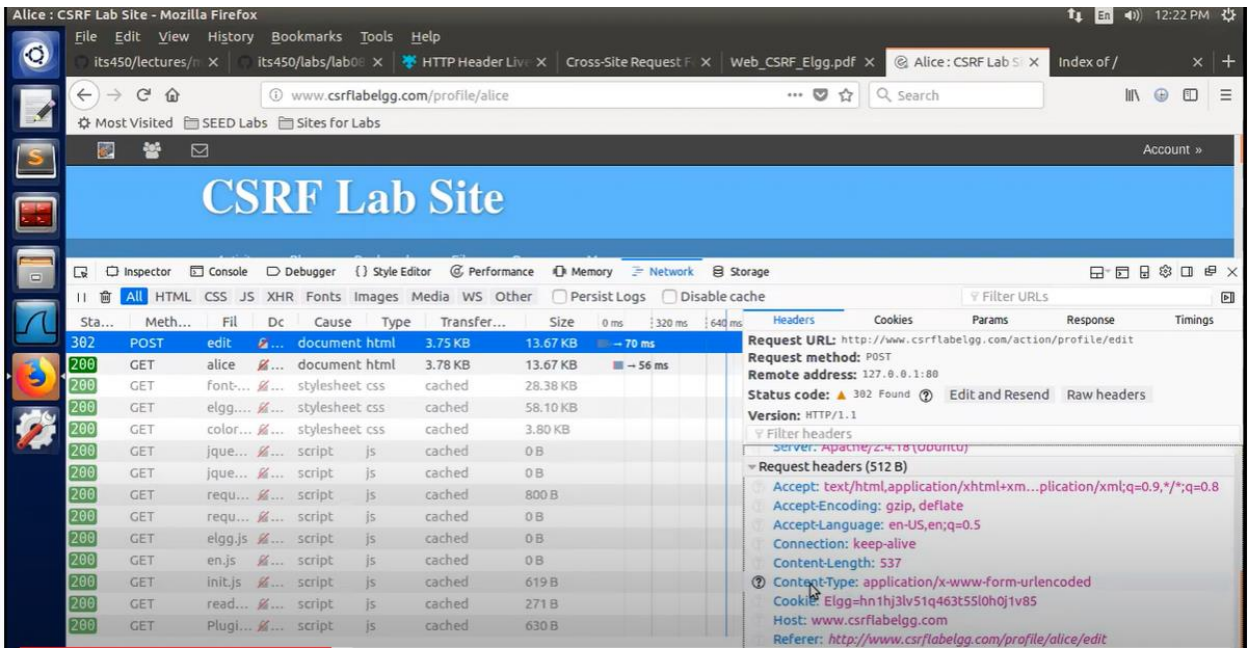


Then, we show some similar information as we use that http header and set then reset the http requests.

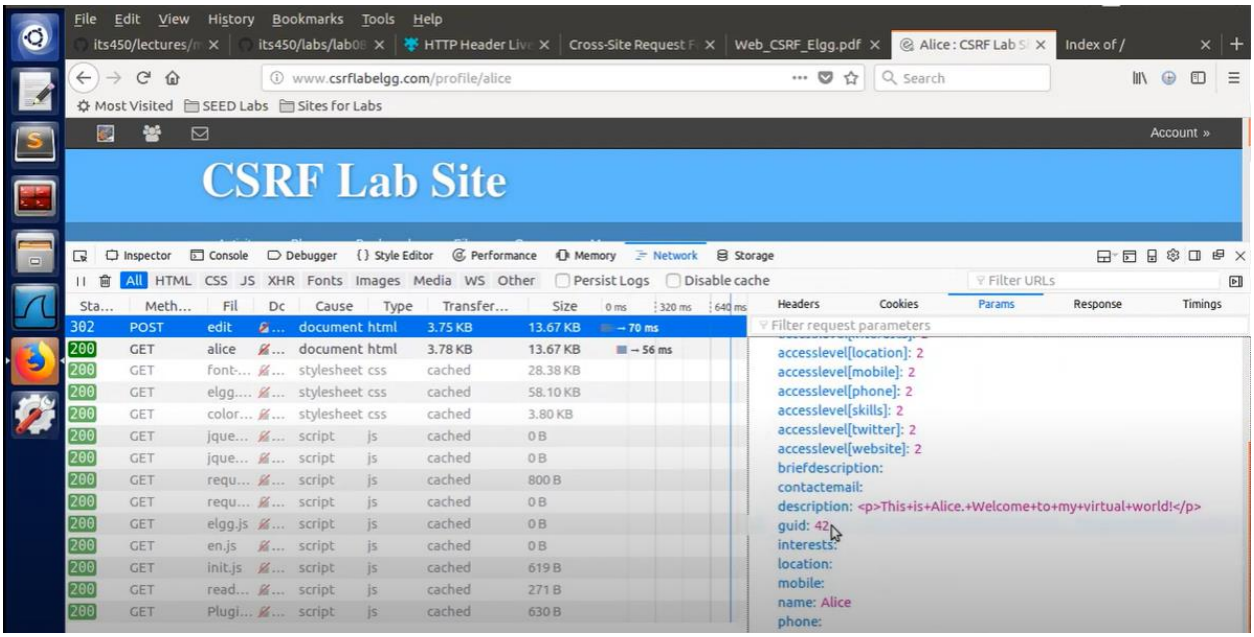
Then we will get 200 http request on the alice profile server.







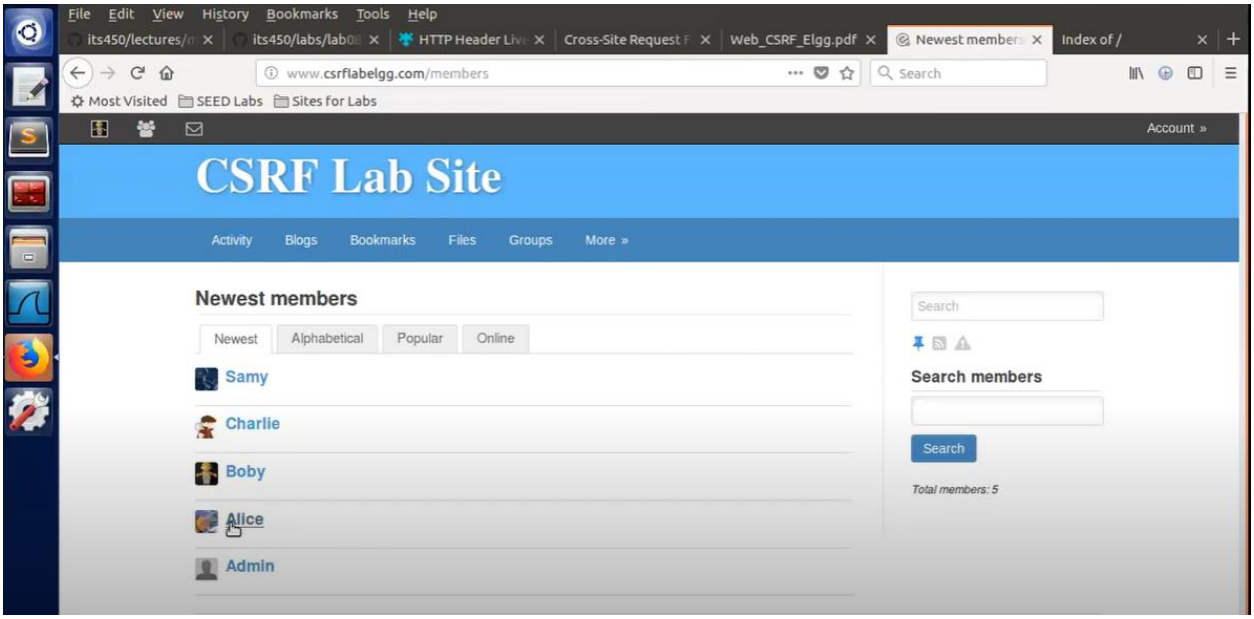
Then we will 200 ok server on alice profile, and got it information as shown below.



Task 2: CSRF Attack using GET Request

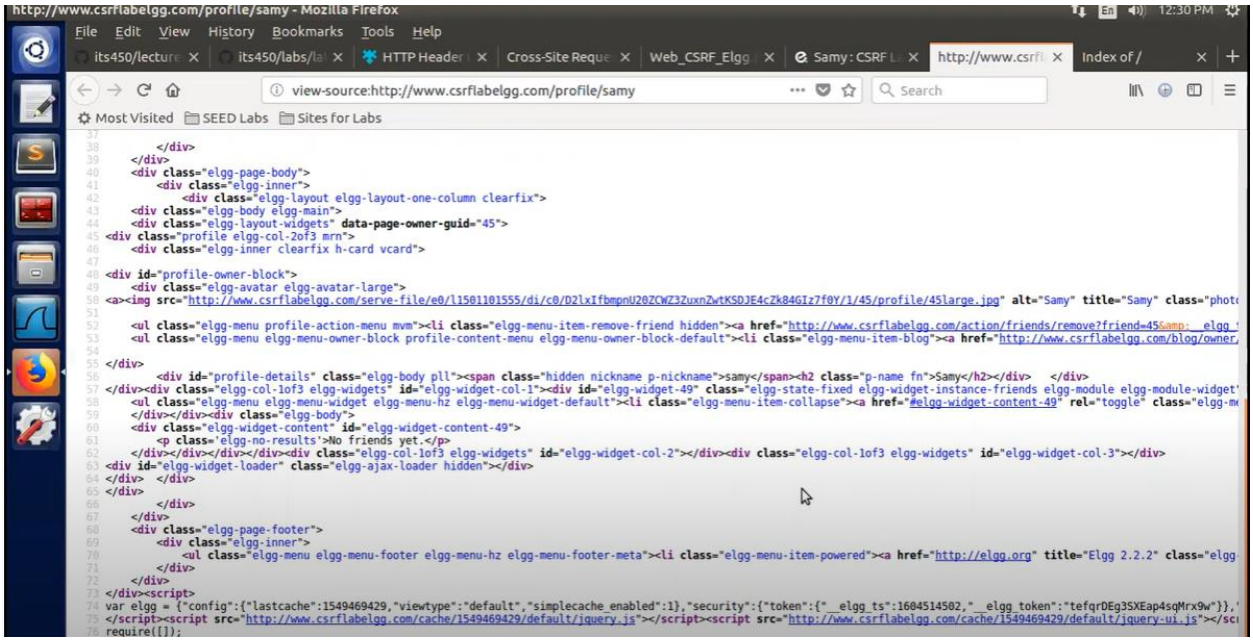
First login csrf scite with Samy username and password too. So, both the friend

Then go to members and 5 members like this.

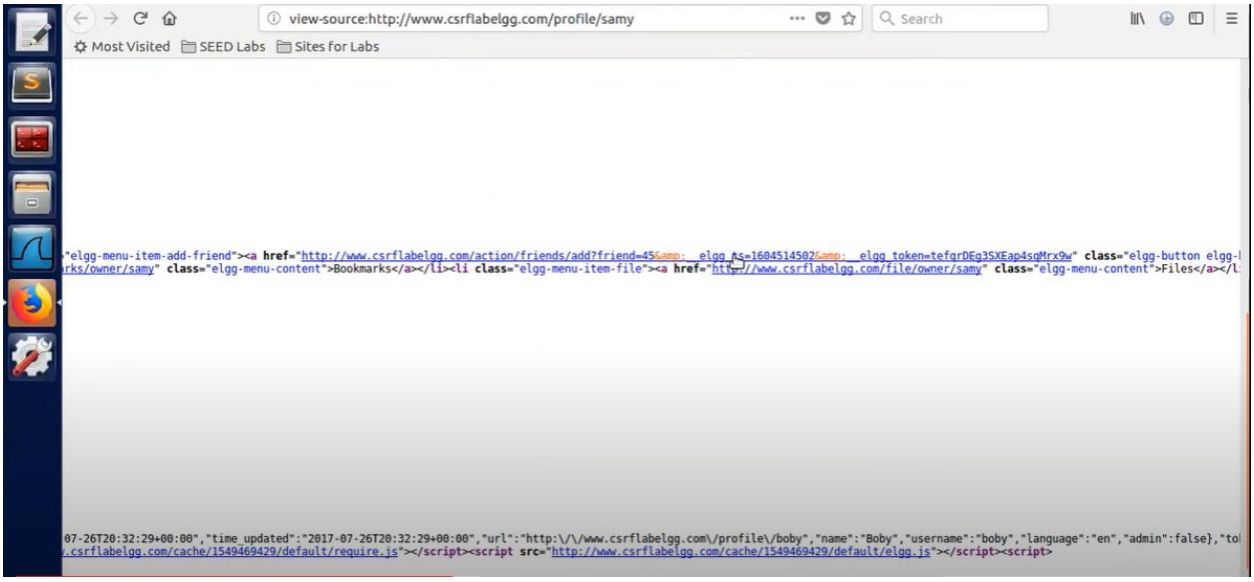


Like alice , samy has its own profile for sending request to server.

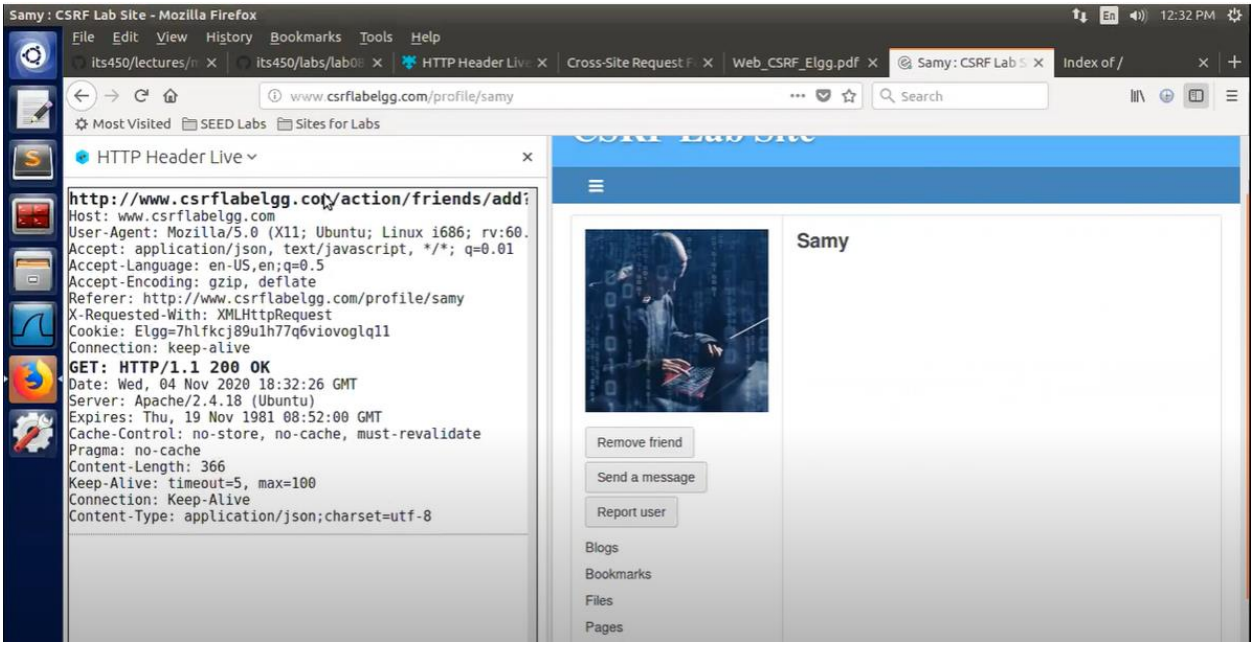
Then, find source code for samy profile, for finding samy cashe id.



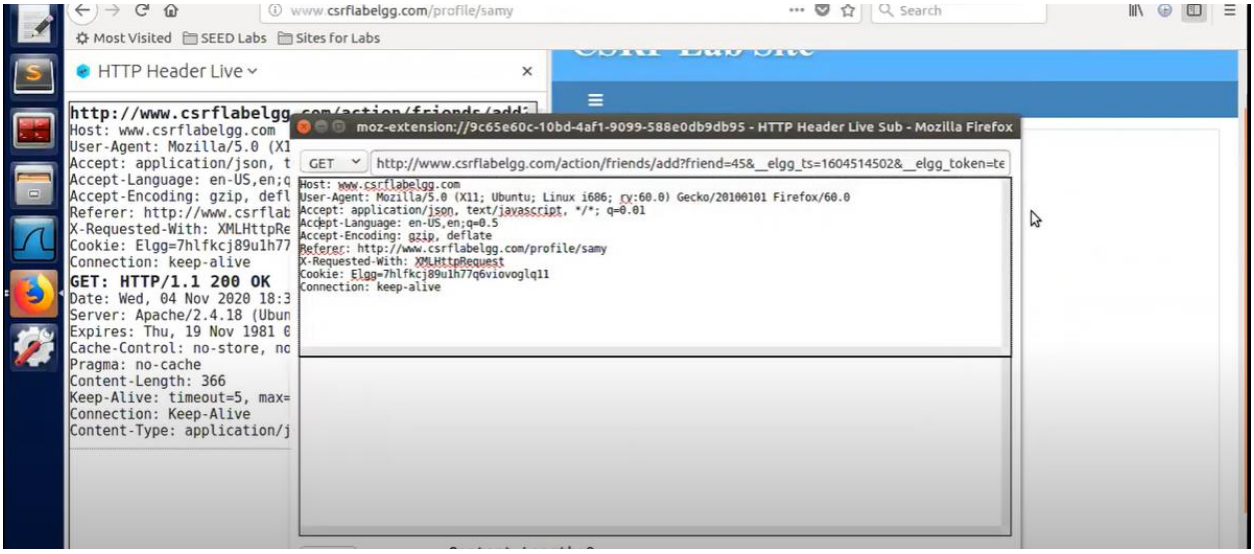
Here is the link to add the friends list as shown below.



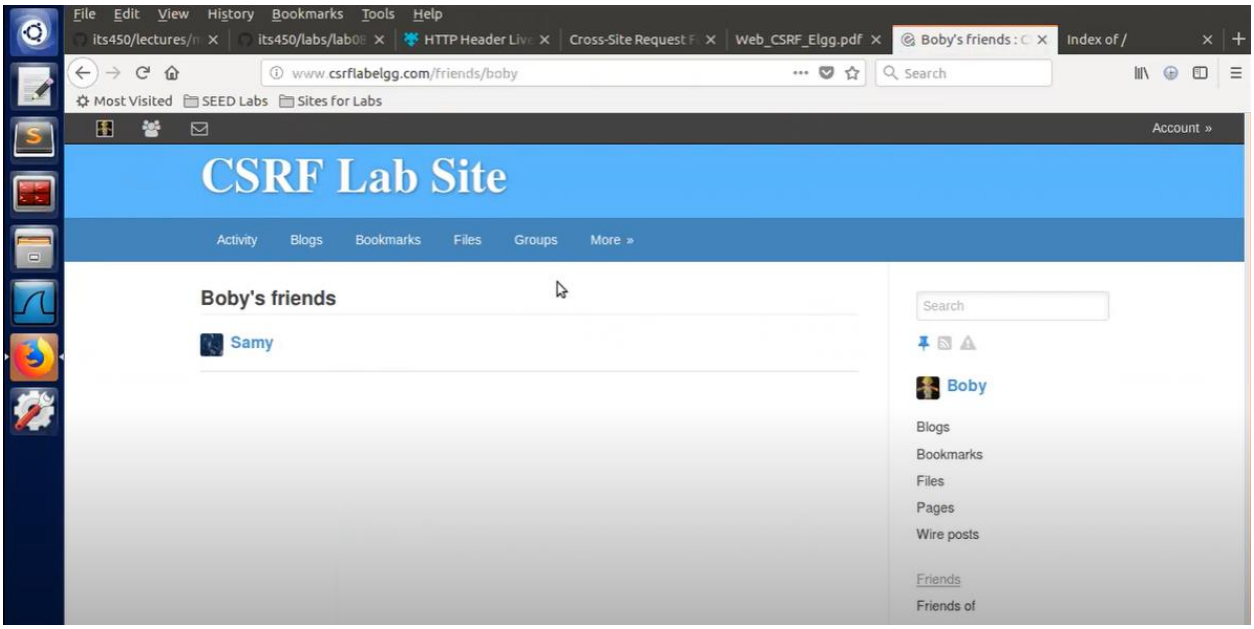
Then we will server responses in http header line ,



The, we will get responses from database, and some http requests.

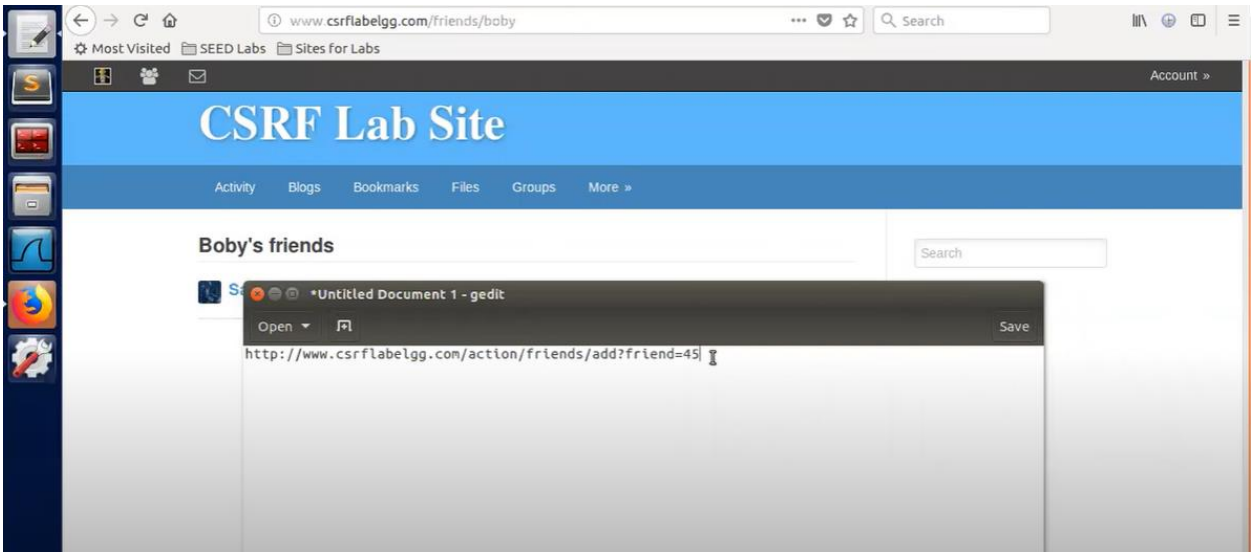


Then we will get , alice have added samy as a friend in its friends list.



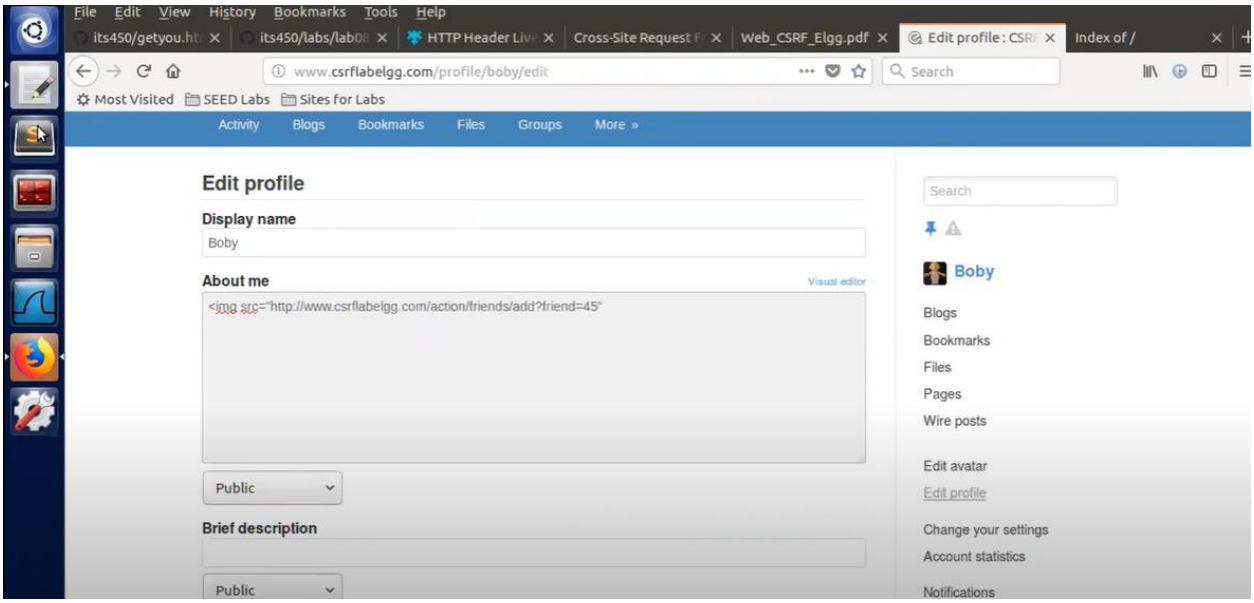
.

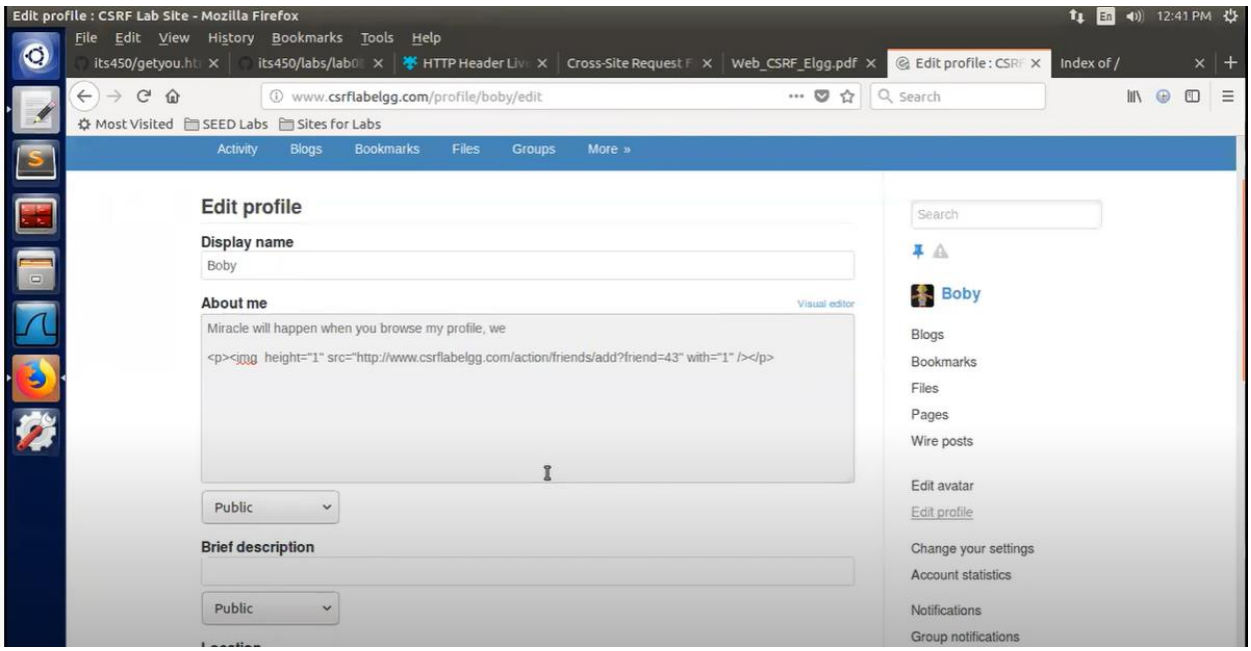
So, we will get the add friend link.



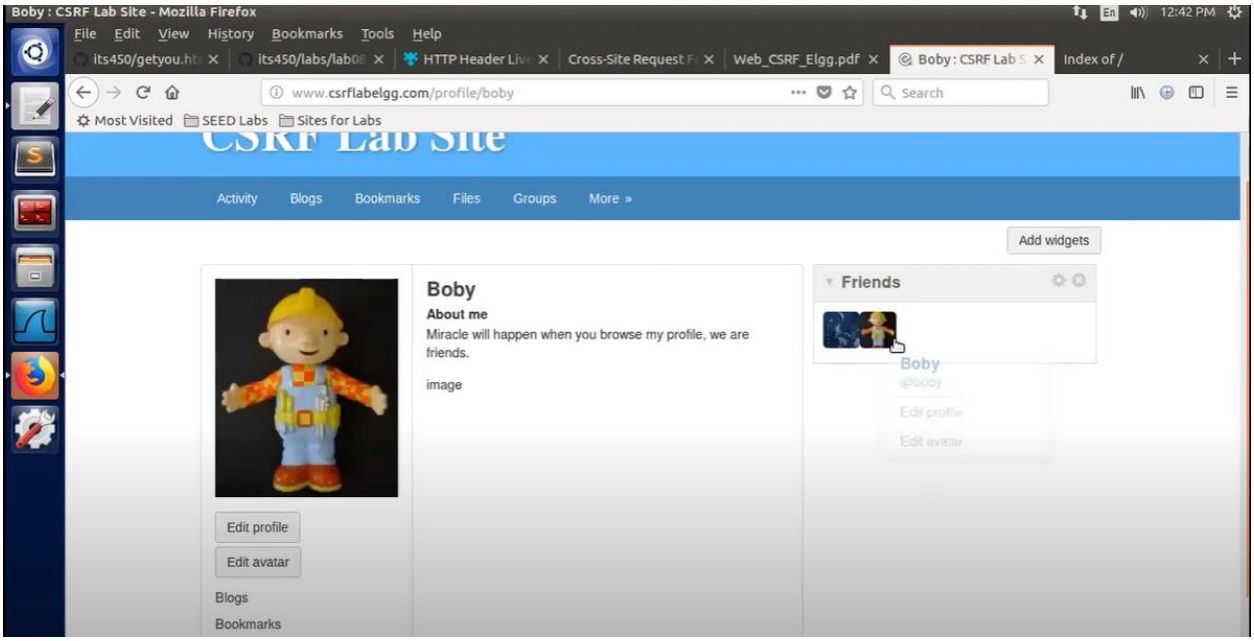
So, in the alice profile, samy shows as in a friend list.

Then we will add this above link in edit profile link.





Like this, friends shows in friend list.

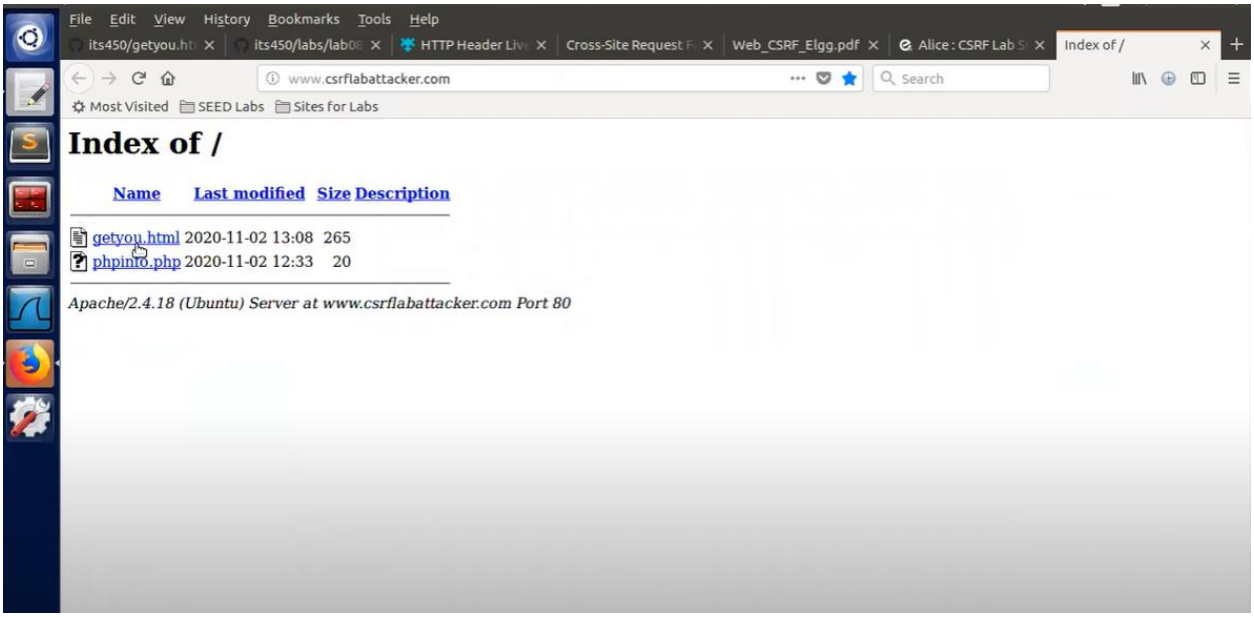


Now, check in alice profile by logging it again after logout from samy, and boby.

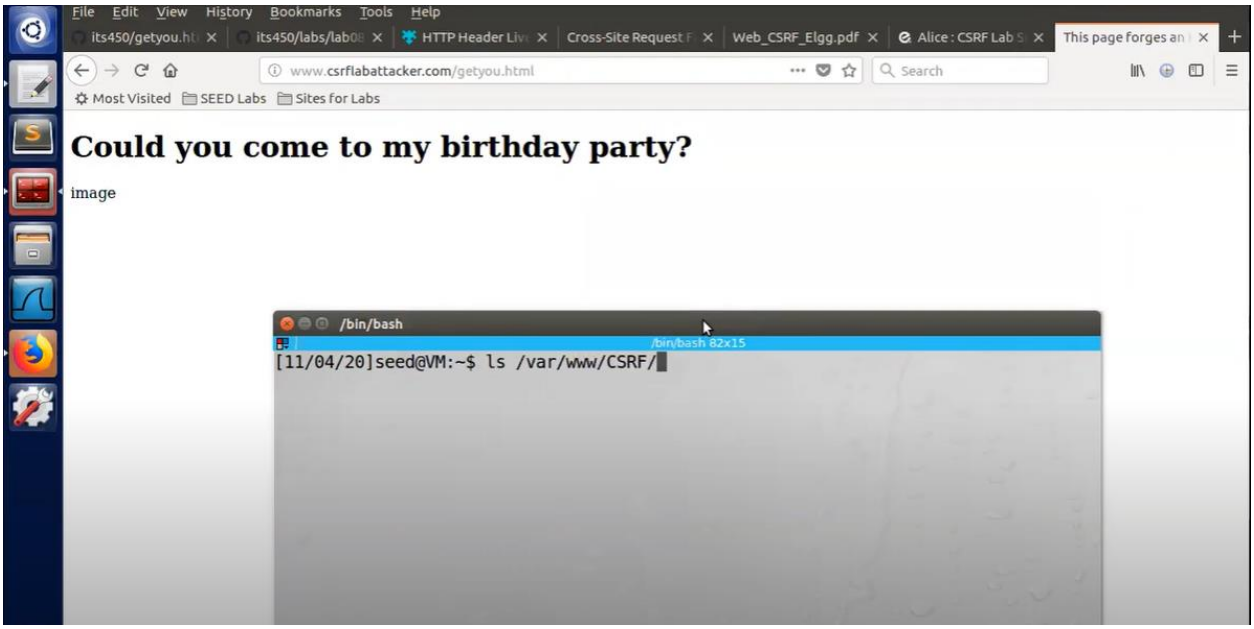
Task 3: CSRF Attack using POST Request

Deliverable 1:

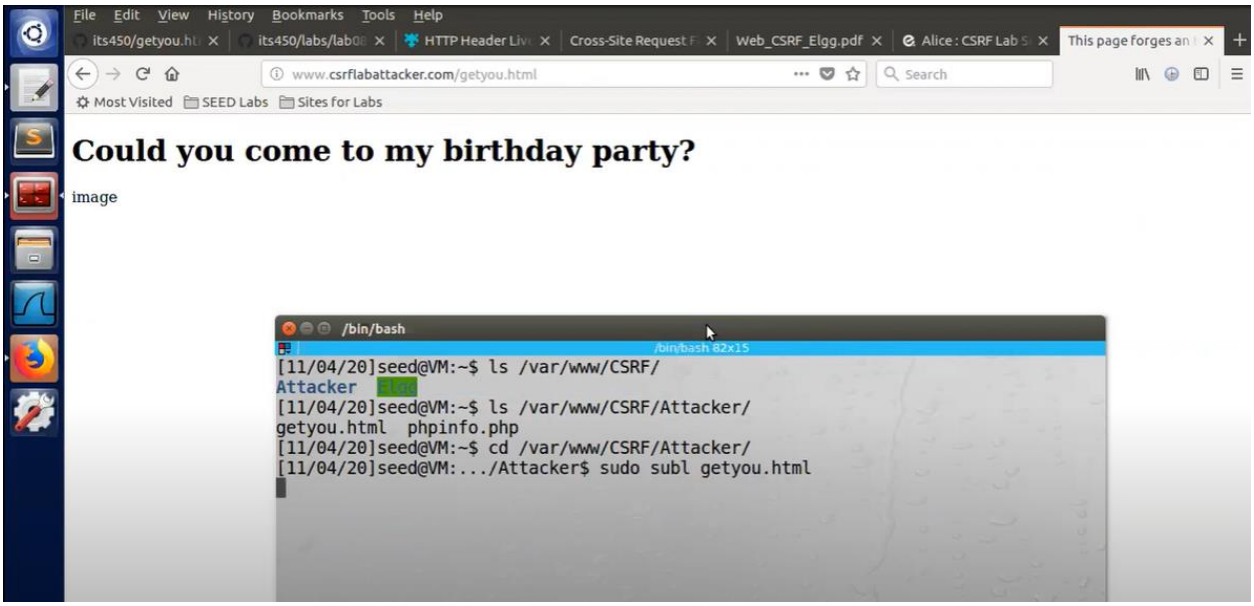
First we will open the server by opening inderof/



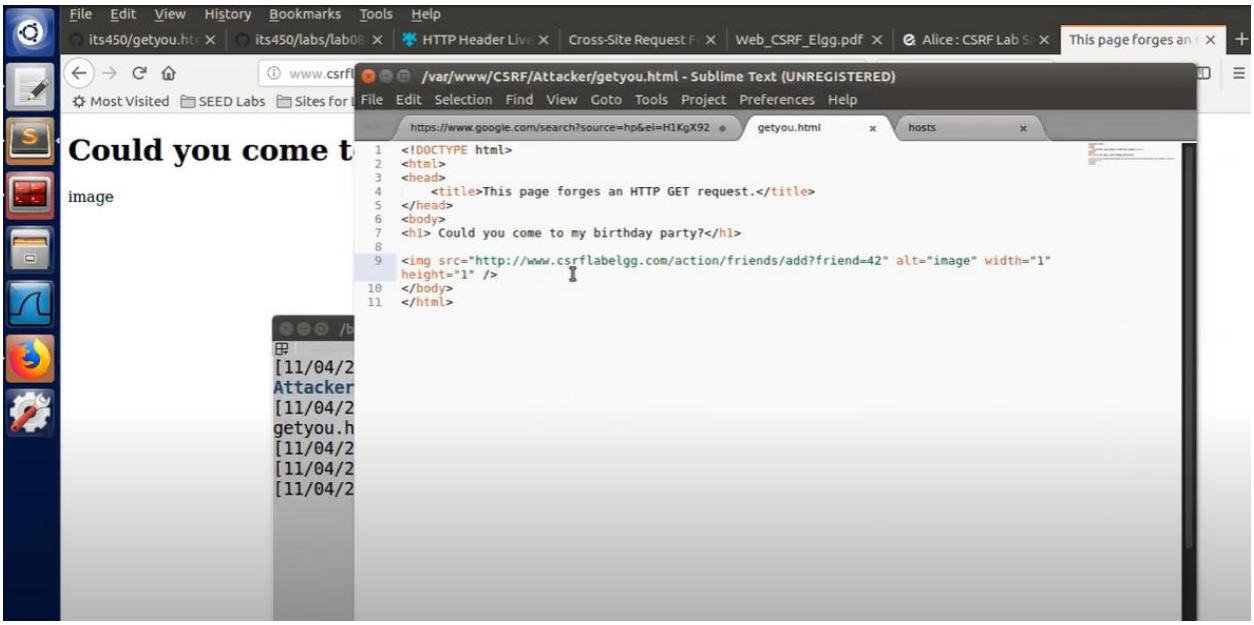
Then, we will add some commands ,getyou.html



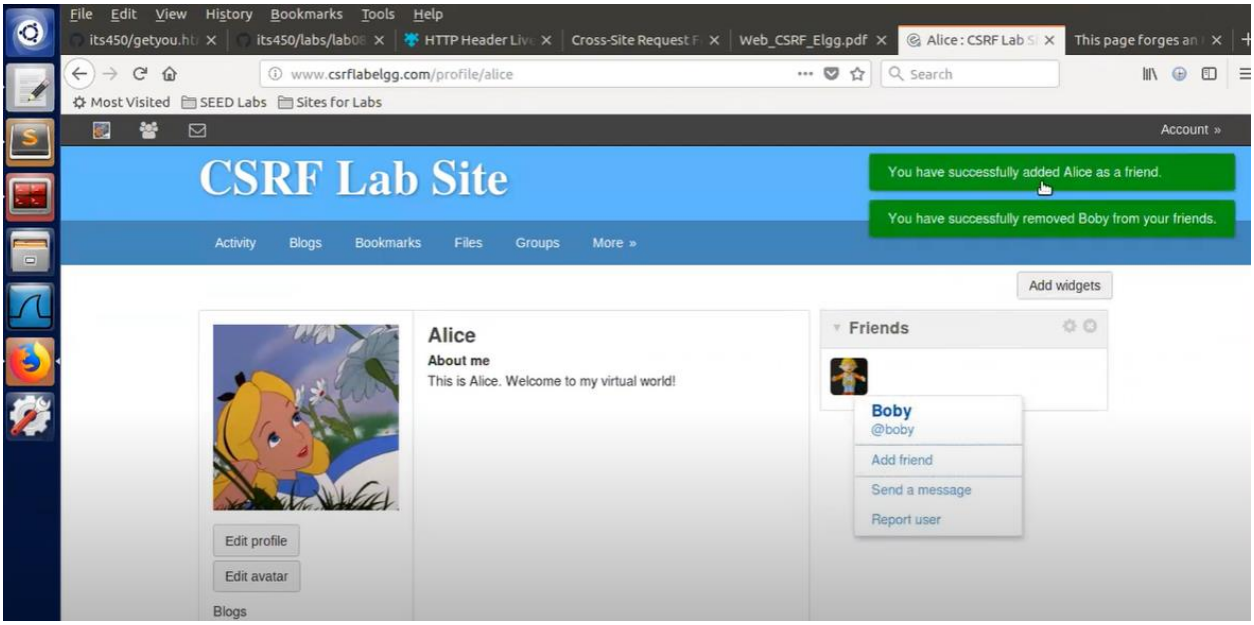
Then, I have written some commands that is given in description.



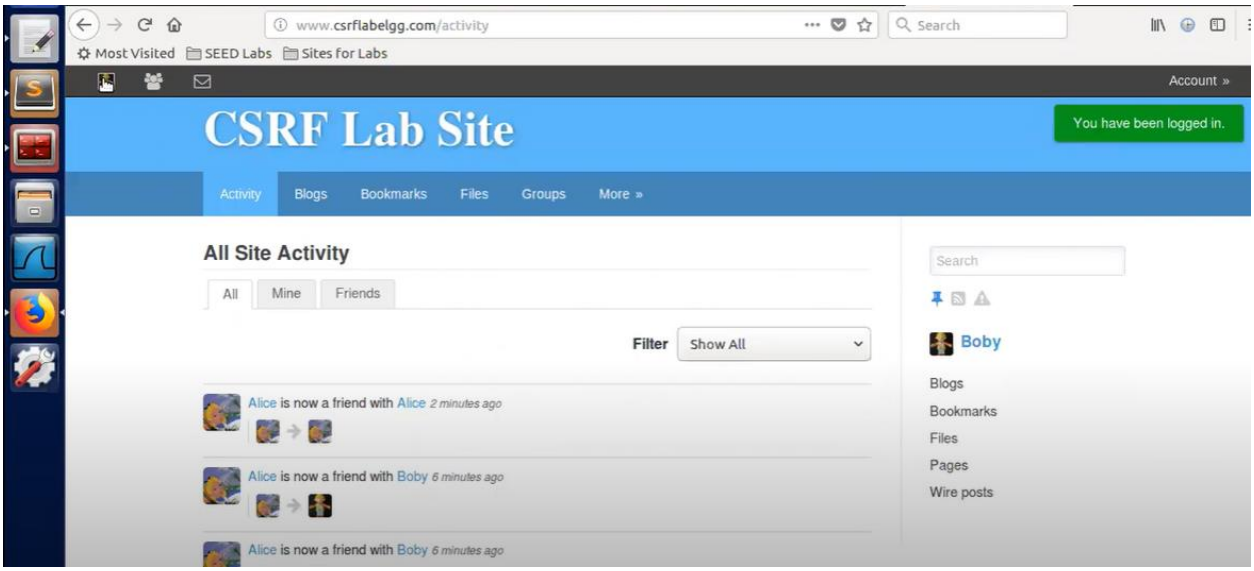
Then, html source code will open.



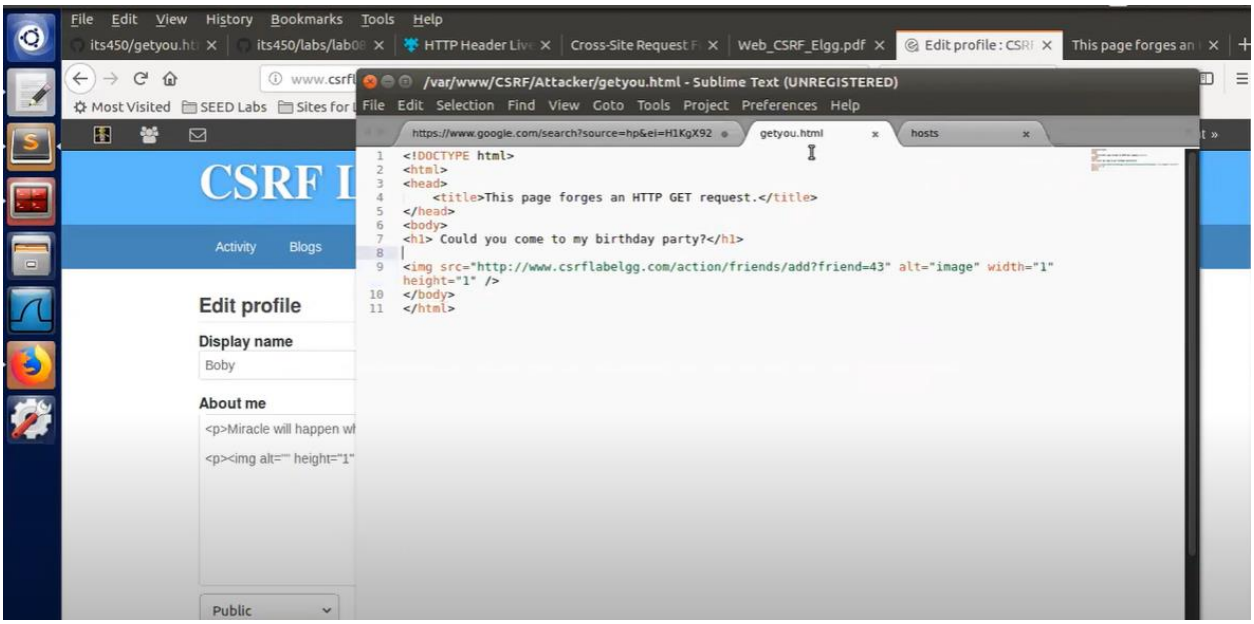
Now, deleted add friend list from friends list.

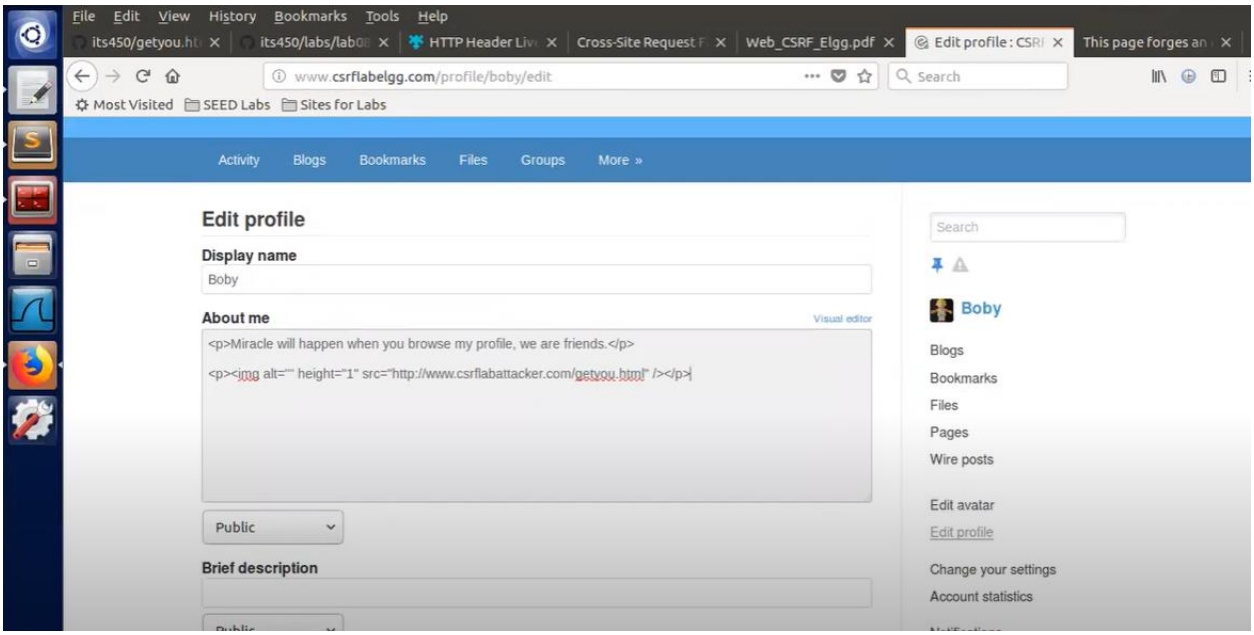


These all are activities of profiles that are shown in csrf site.



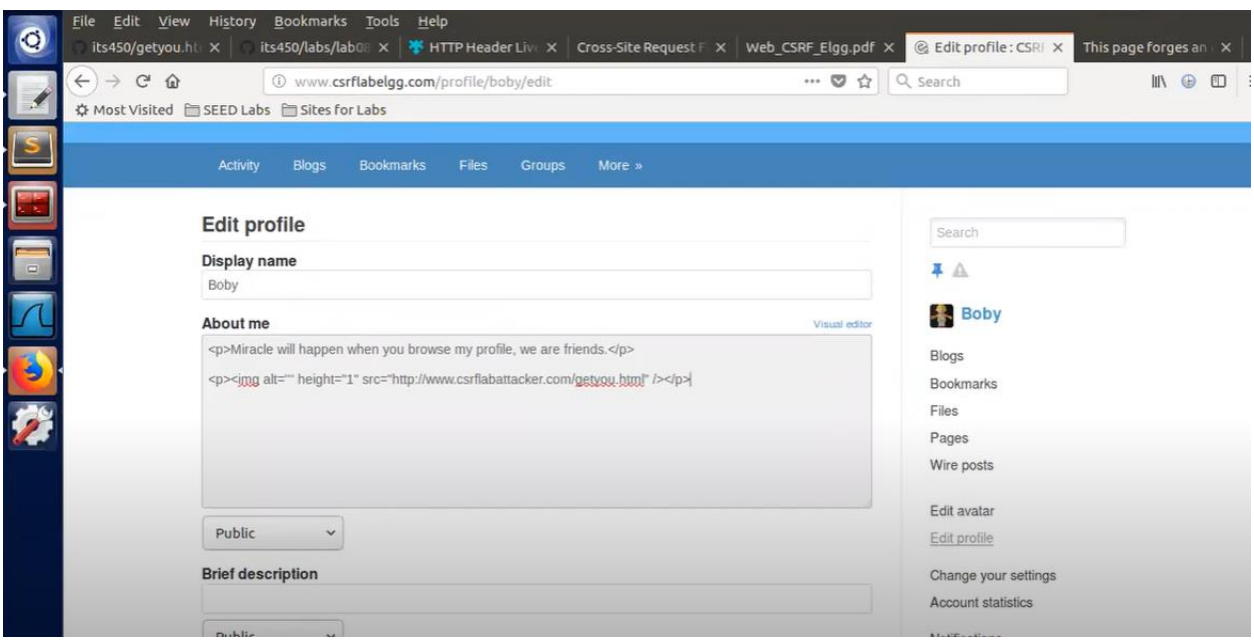
Deliverable 2:





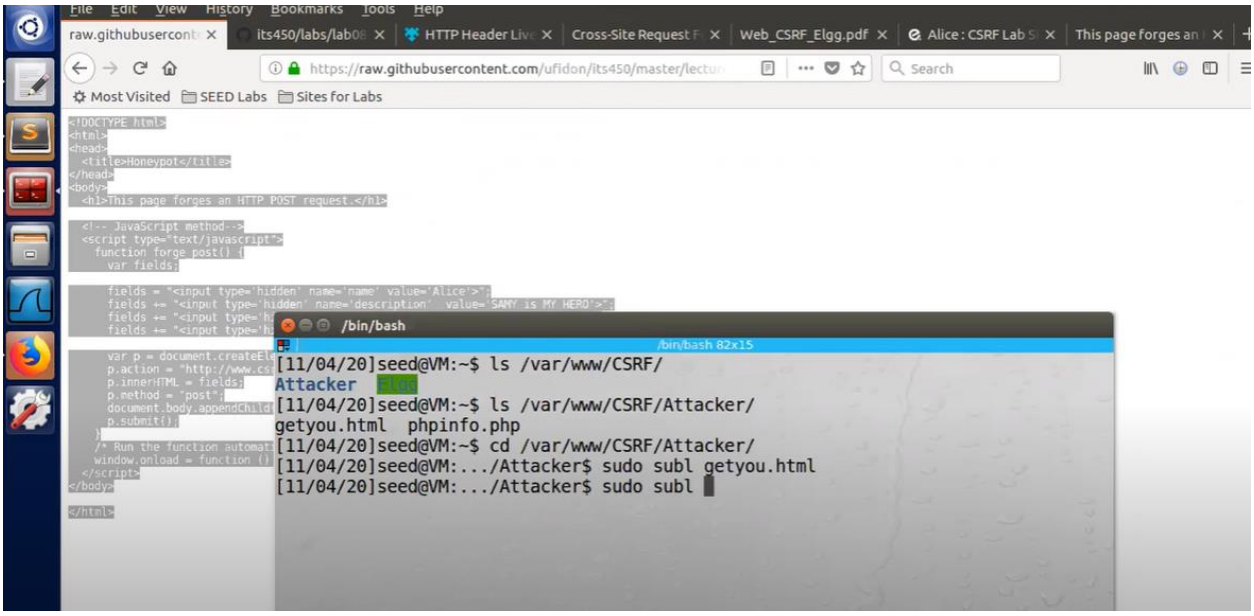
Deliverable 3:

For this we have tologin alice profile, that will help to log in others profile too.

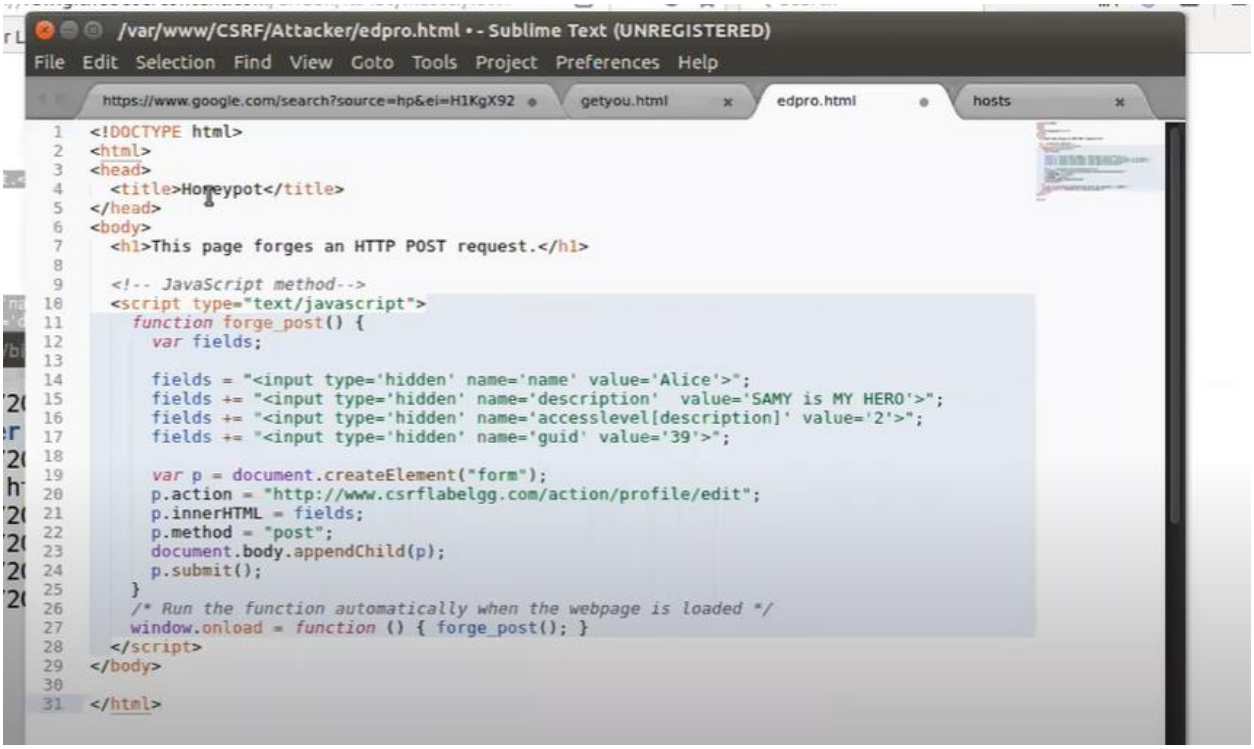


Deliverable 4:

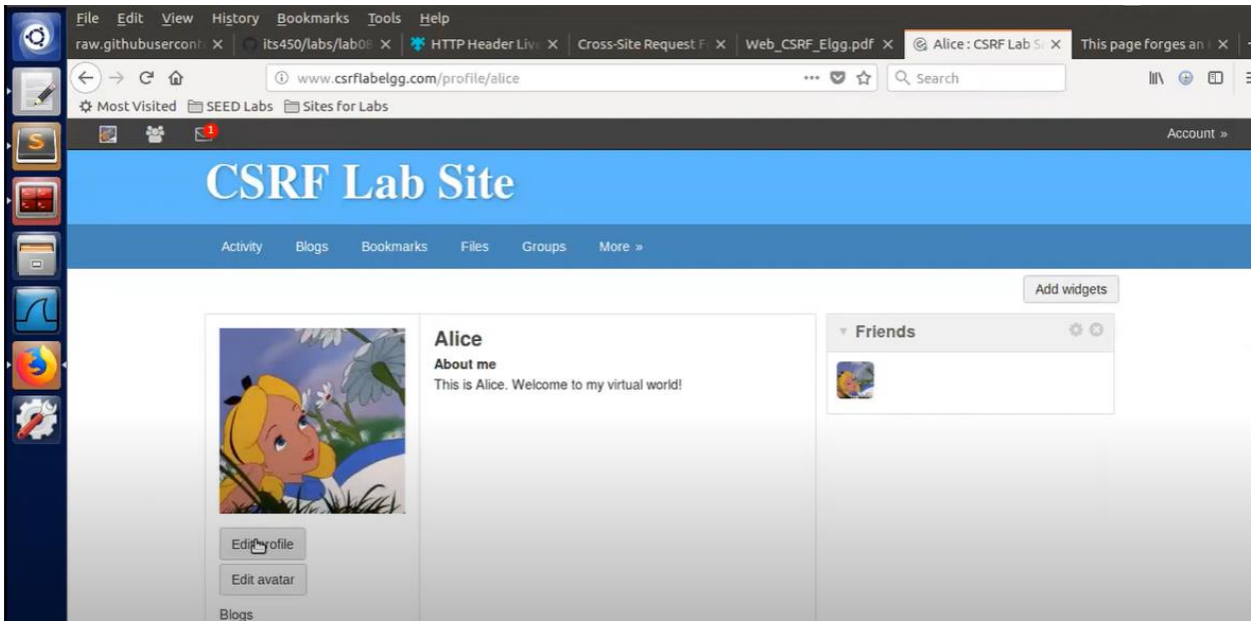
Add the source code to sublime of VM by using sudo subl edpro.html command in bash.



Then , this sublime screen will come up

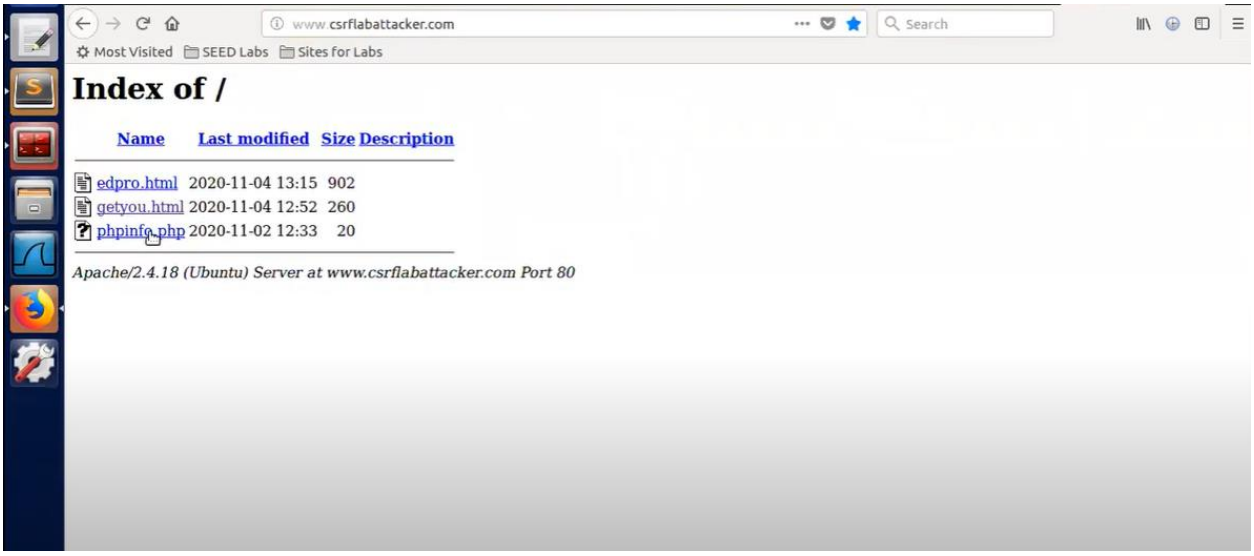


Then check our changes in alice profile

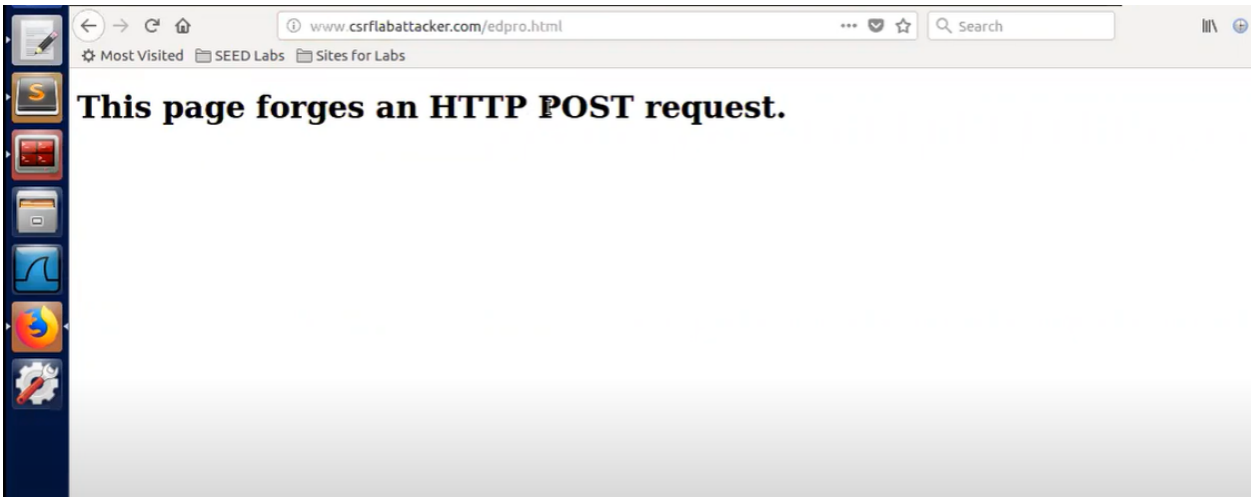


Then search www.csrflab-attacker.com. In search engine.

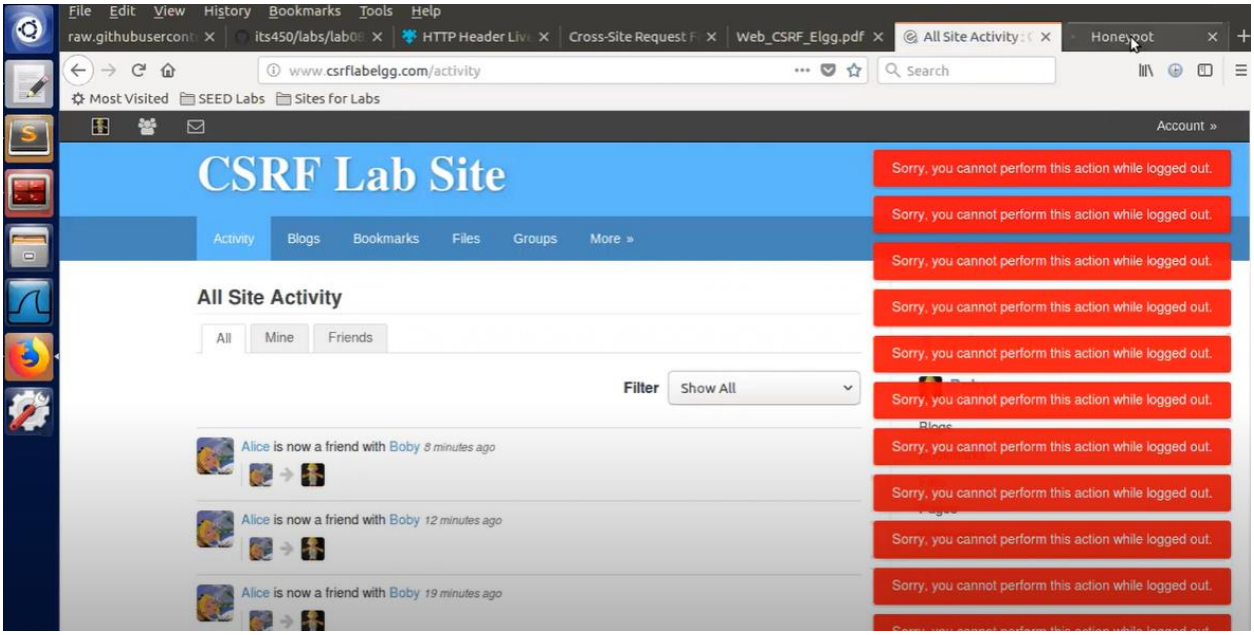
We will get this



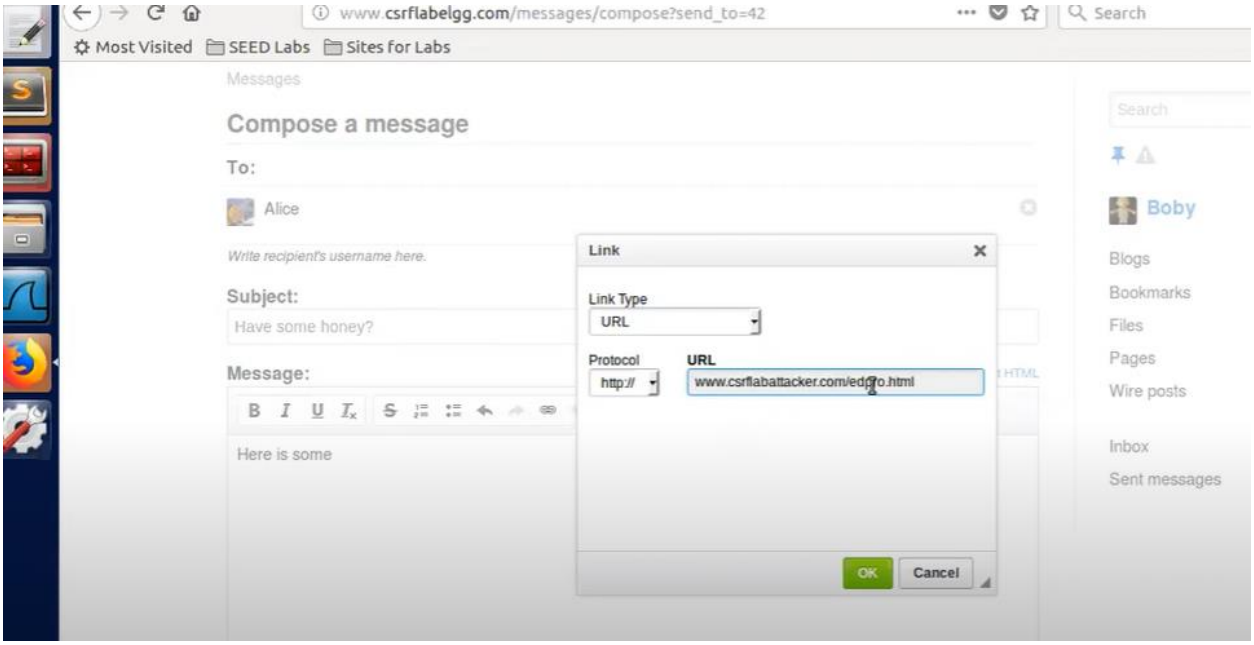
After clicking on edpro.html. This will open up.

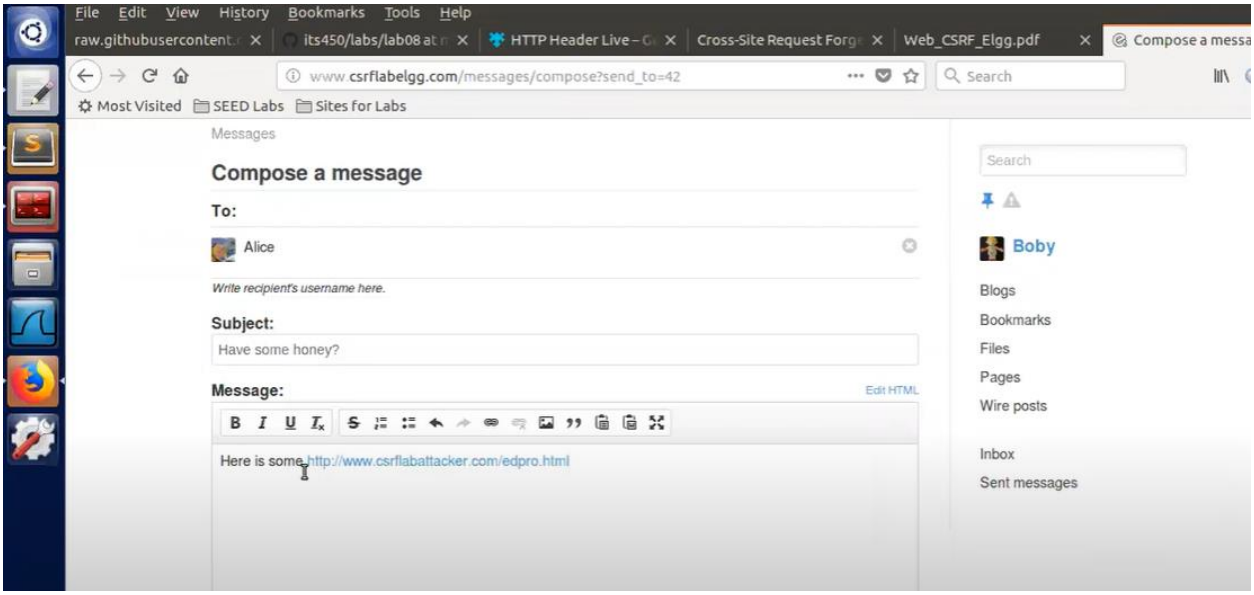


On the csrf scite, the all profile are logged off due to this activity.

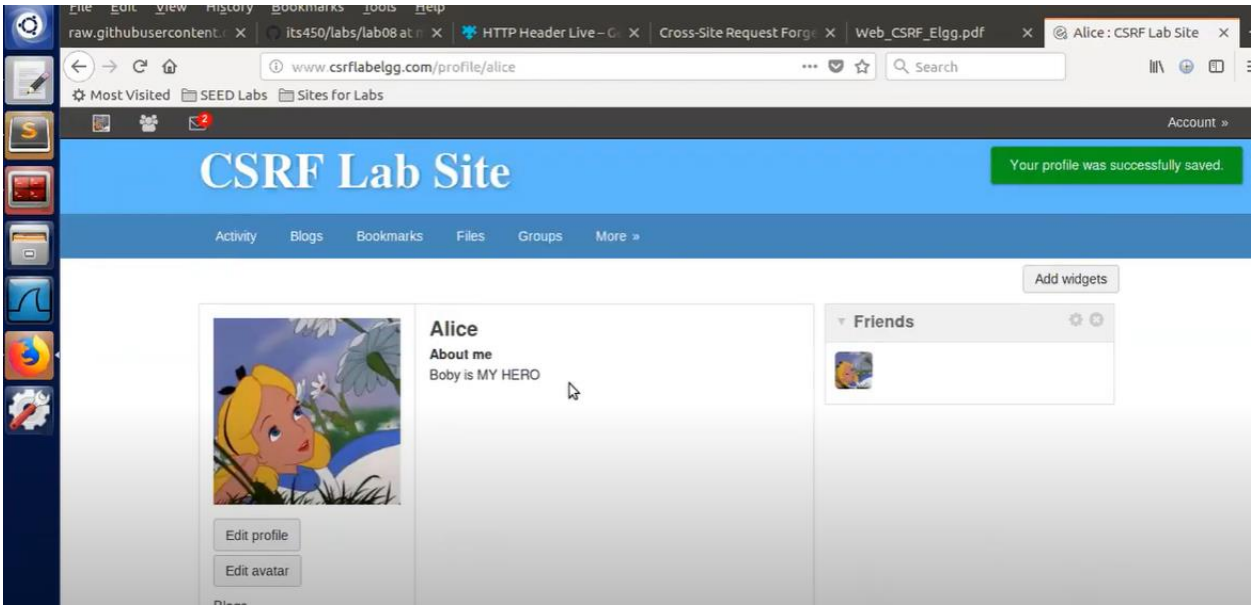


Then we will http server post link



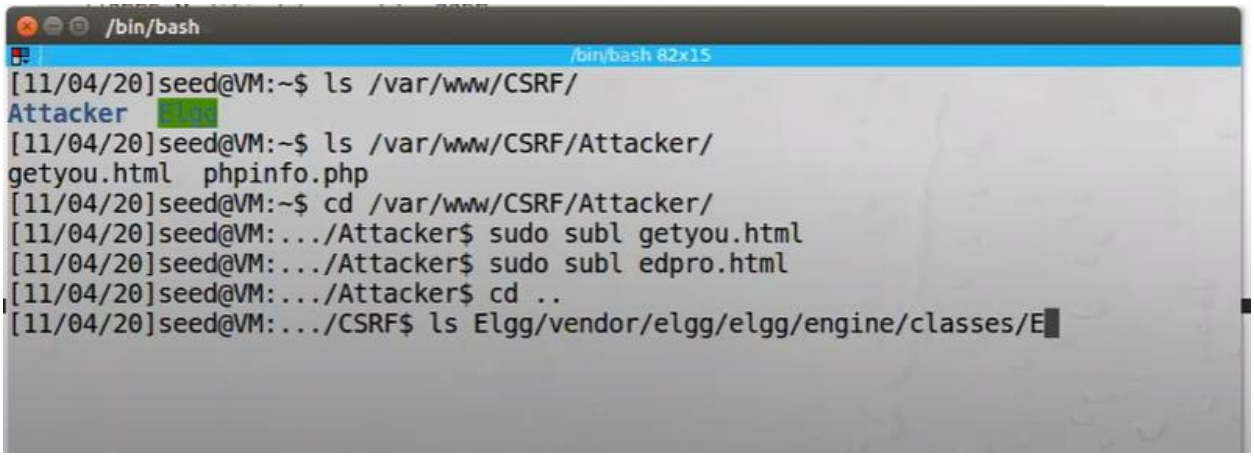


Then this will see in the alice profile.



Task 4: Enabling Elgg's Countermeasure

Type all the commands on bash that is written on requirement sheet,



Then paste the link on sublime of ubuntu

/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg/ActionsService.php - Sublime Text (UNREGISTERED)

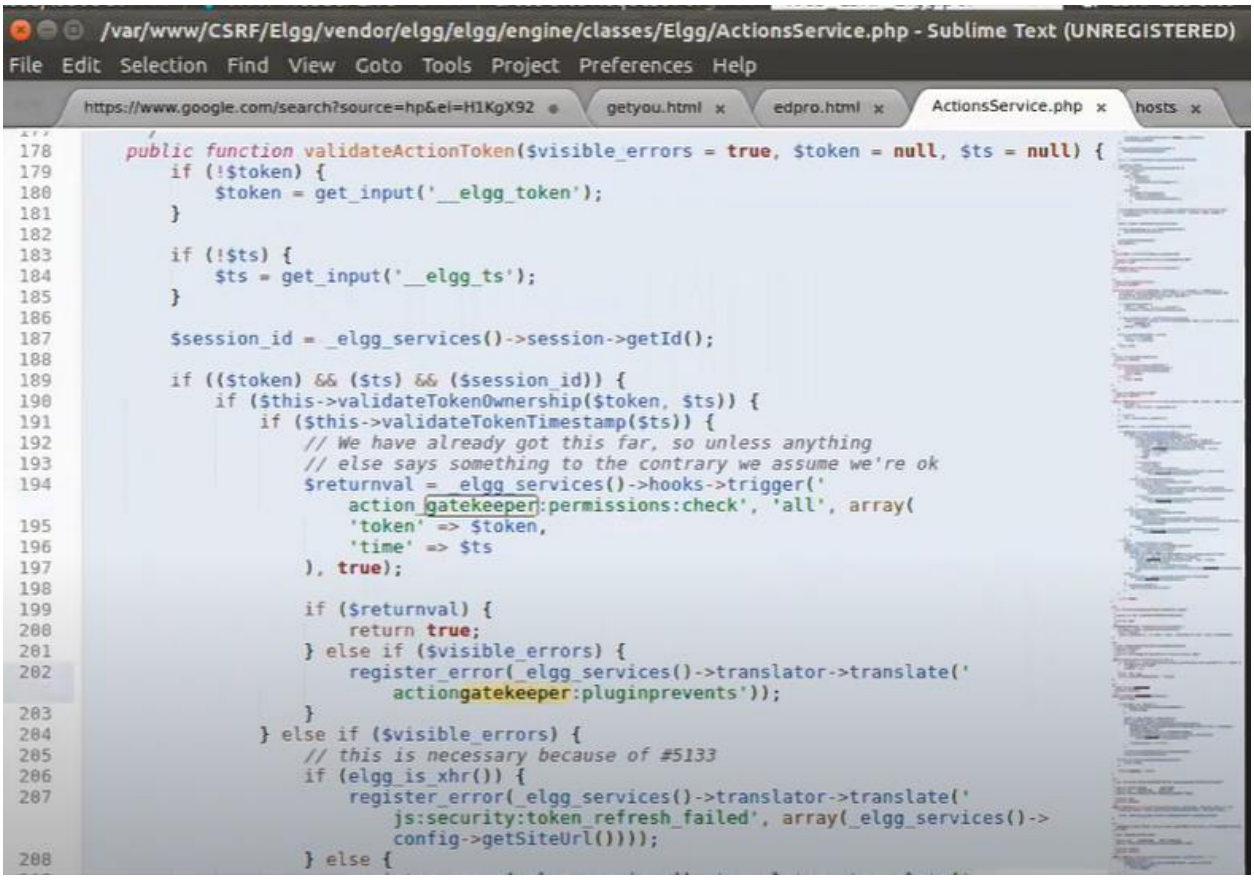
File Edit Selection Find View Goto Tools Project Preferences Help

https://www.google.com/search?source=hp&ei=H1KgX92 | getyou.html | edpro.html | ActionsService.php | hosts

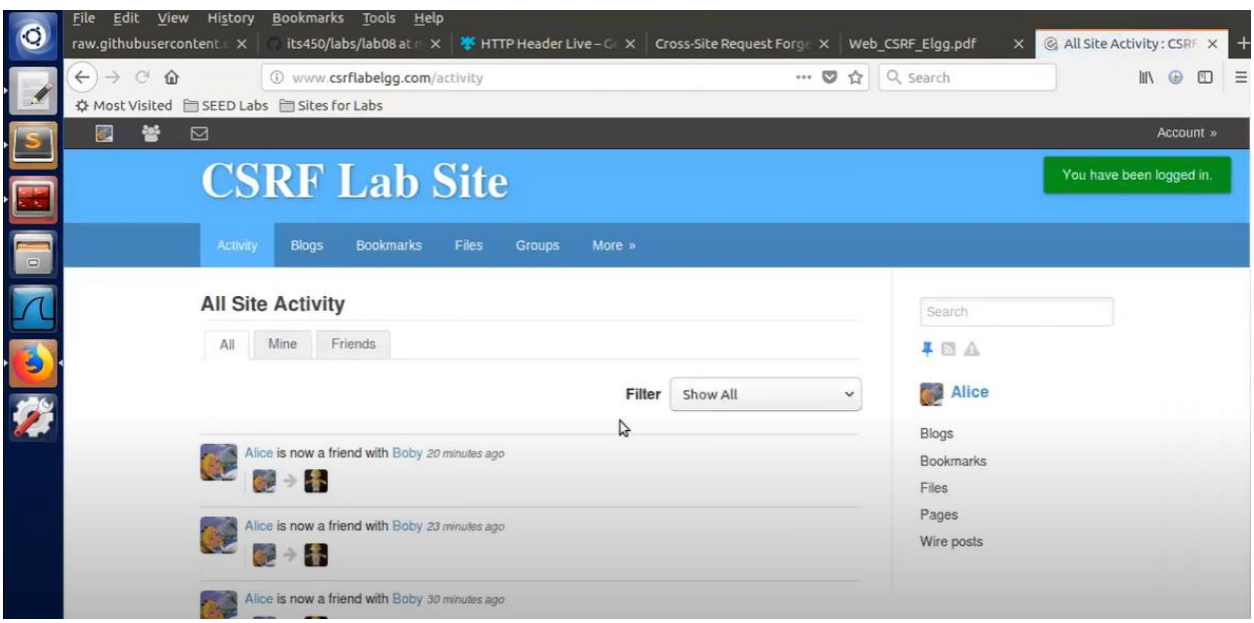
```php
<?php
namespace Elgg;
use Elgg\Services\AjaxResponse;

/**
 * WARNING: API IN FLUX. DO NOT USE DIRECTLY.
 *
 * Use the elgg_* versions instead.
 *
 * @access private
 *
 * @package    Elgg.Core
 * @subpackage Actions
 * @since      1.9.0
 */
class ActionsService {

    /**
     * Registered actions storage
     *
     * Each element has keys:
     *   "file" => filename
     *   "access" => access level
     *
     * @var array
     */
    private $actions = array();

    /**
     * The current action being processed
     * @var string
     */
    private $currentAction = null;
```

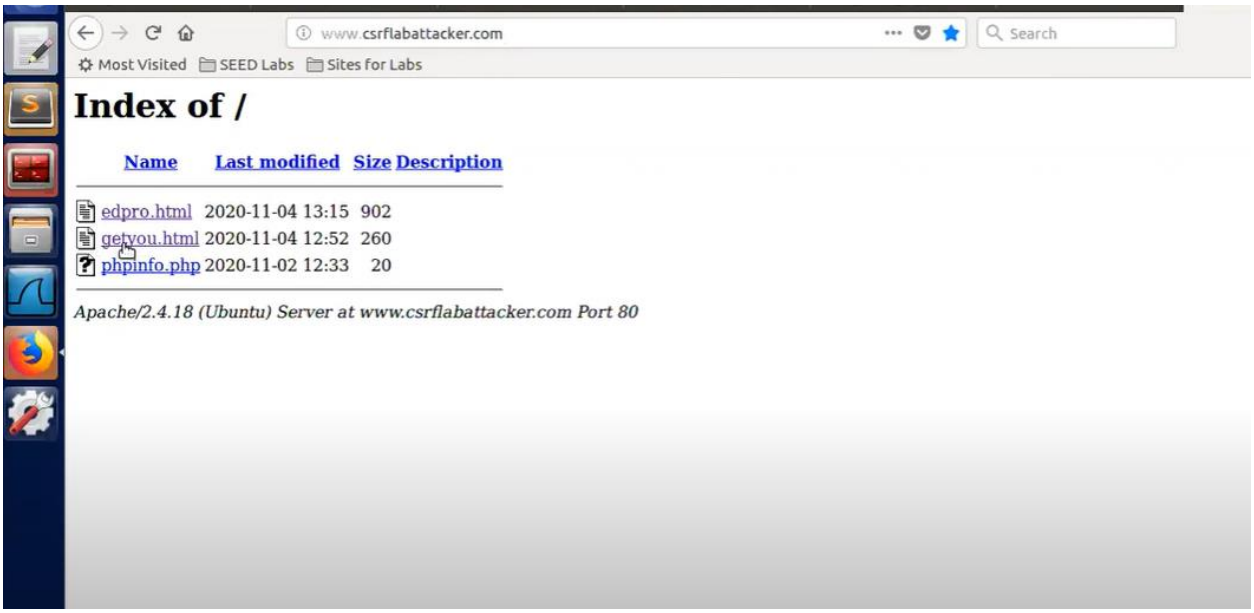
https://www.google.com/search?source=hp&ei=H1KgX92 | getyou.html | edpro.html | ActionsService.php | hosts

```php
    public function execute($action, $forwarder = "") {
        $action = rtrim($action, '/');
        $this->currentAction = $action;

        // @todo REMOVE THESE ONCE #1589 IS IN PLACE.
        // Allow users to disable plugins without a token in order to
        // remove plugins that are incompatible.
        // Login and logout are for convenience.
        // file/download (see #2010)
        $exceptions = array(
            'admin/plugins/disable',
            'logout',
            'file/download',
        );

        if (!in_array($action, $exceptions)) {
            // All actions require a token.
            $this->gatekeeper($action);
        }

        $forwarder = str_replace(_elgg_services()->config->getSiteUrl(), "", $forwarder);
        $forwarder = str_replace("http://", "", $forwarder);
        $forwarder = str_replace("@", "", $forwarder);
        if (substr($forwarder, 0, 1) == "/") {
            $forwarder = substr($forwarder, 1);
        }

        /**
         * Complete the execution with a forward
         *
         * @param string $error_key Error message key
         *
         * @throws \SecurityException
         */
```

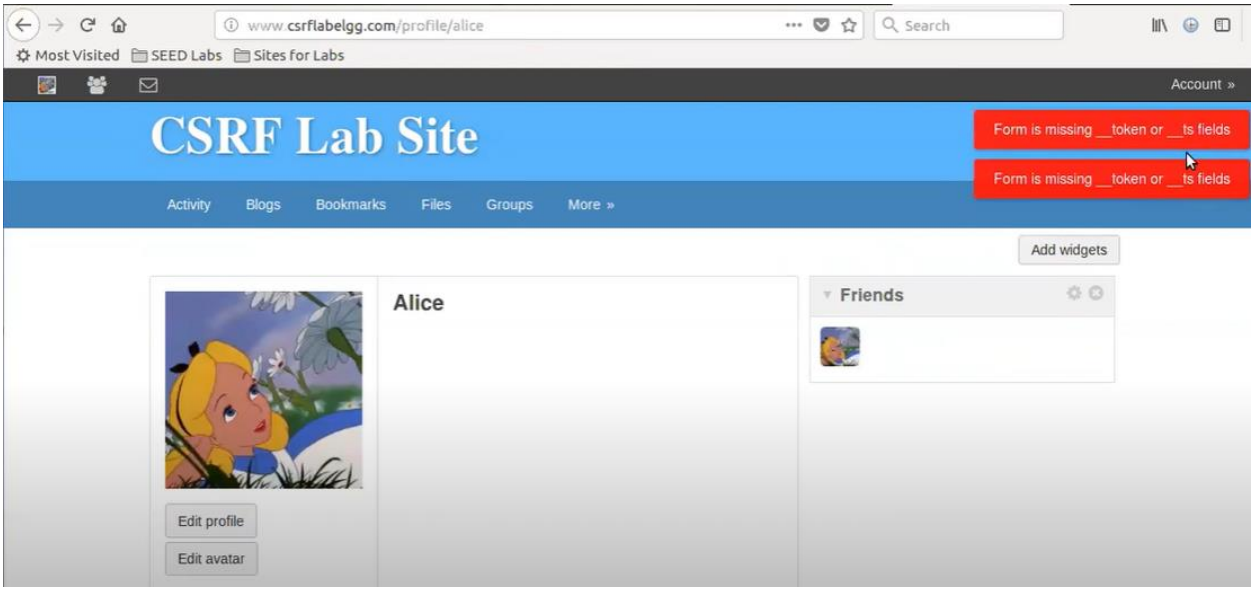


Then login csrf login page with alice username.



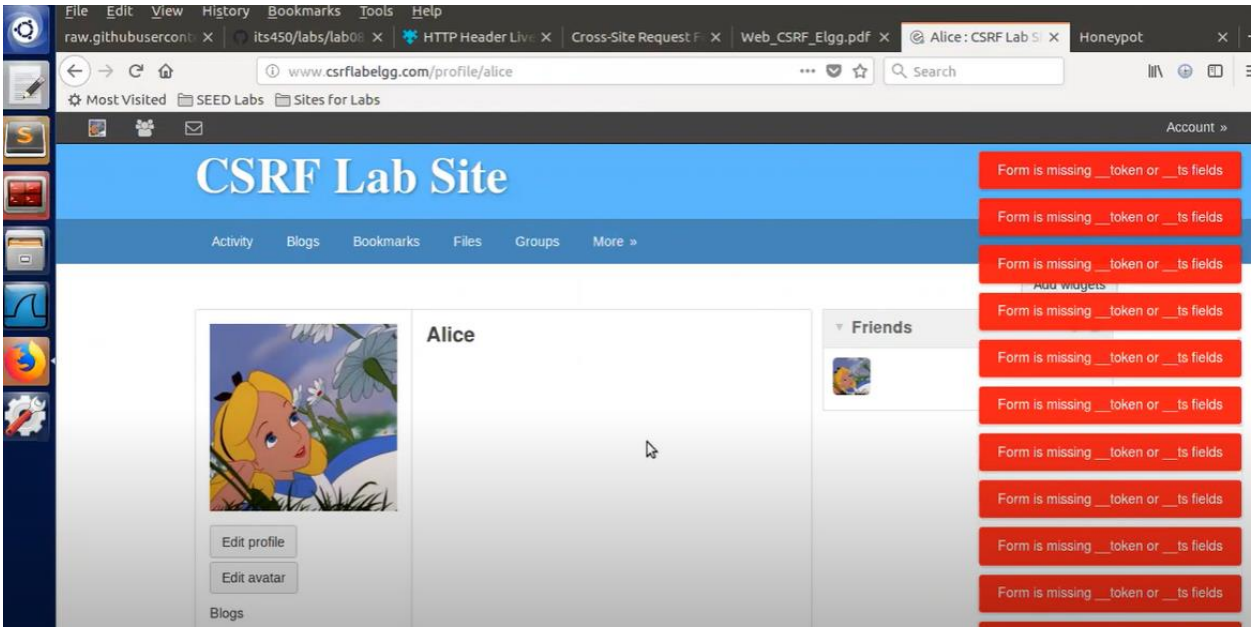


Then, in alice account send some validations of tokens and shows some red alerts.

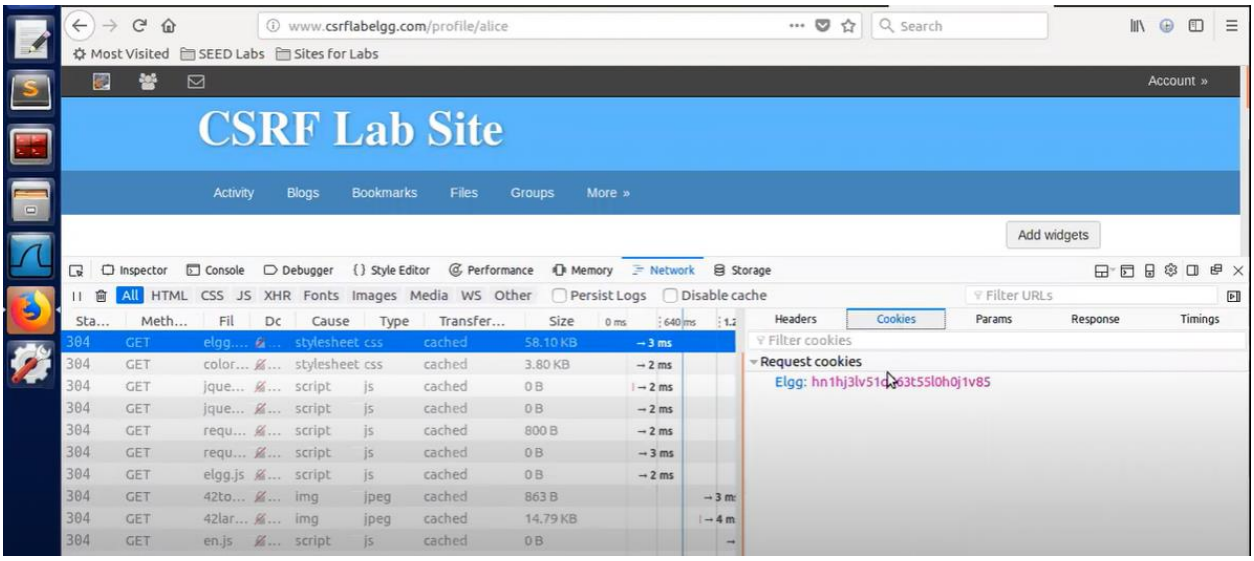


Task 5: Experimenting with the SameSite Cookie Method

For finding cookies, the alice profile shows many of the validations to the attacker and user.



Upon sending an HTTP ask (authentic or something else), the victim's browser will incorporate the cookie header. Treats are regularly utilized to store a user's session identifier so that the client does not have to be enter their login qualifications for each ask, which would clearly be illogical. On the off chance that the victim's verification session is put away in a session cookie that's still substantial (a browser window/tab does not fundamentally ought to be open), and in the event that the application is helpless to Cross-site Request Forgery (CSRF), at that point the aggressor can use CSRF to dispatch any craved pernicious demands against the site and the server-side code is incapable to recognize whether these are authentic demands.



The Same Site cookie property may be a unused quality that can be set on treats to educated the browser to debilitate third-party utilization for particular treats. The Same Site quality is set by the server when setting the cookie and demands the browser to as it were send the cookie in a first-party setting. Hence, the ask has got to start from the same root – demands made by third-party destinations will not incorporate the Same Site cookie. This successfully dispenses with Cross-site Ask Imitation assaults without the utilize of synchronizer tokens.

Set the SameSite quality of your treats to Strict. In the event that this would break your web application usefulness, set the SameSite property to Remiss but never to None. Not all browsers bolster SameSite treats however, but most do. Utilize this property as extra assurance along side anti-CSRF tokens