CSCI 5234 Web Security

Lab2

Cross-Site Scripting (XSS) Attack

Lab Environment:

1. Follow the instructions given on the Lab Setup page and Web_XSS_Elgg to download, install, and configure the virtual machines (VMs).
2. The Cross-Site Scripting Attack will have to use two VMs, victim and attacker VMs; or, you can use one VM and login and out for two characters(Please notice, you have to clean cache if you use only one VM).
3. In both VMs, modify the /etc/hosts file to map the domain name of www.xsslabelgg.com to the attacker machine's IP address. Modify 127.0.0.1 to the attacker machine's IP address as shown in Figure 1.
   192.168.0.165 www.csrflabelgg.com
   192.168.0.165 www.csrflabattacker.com

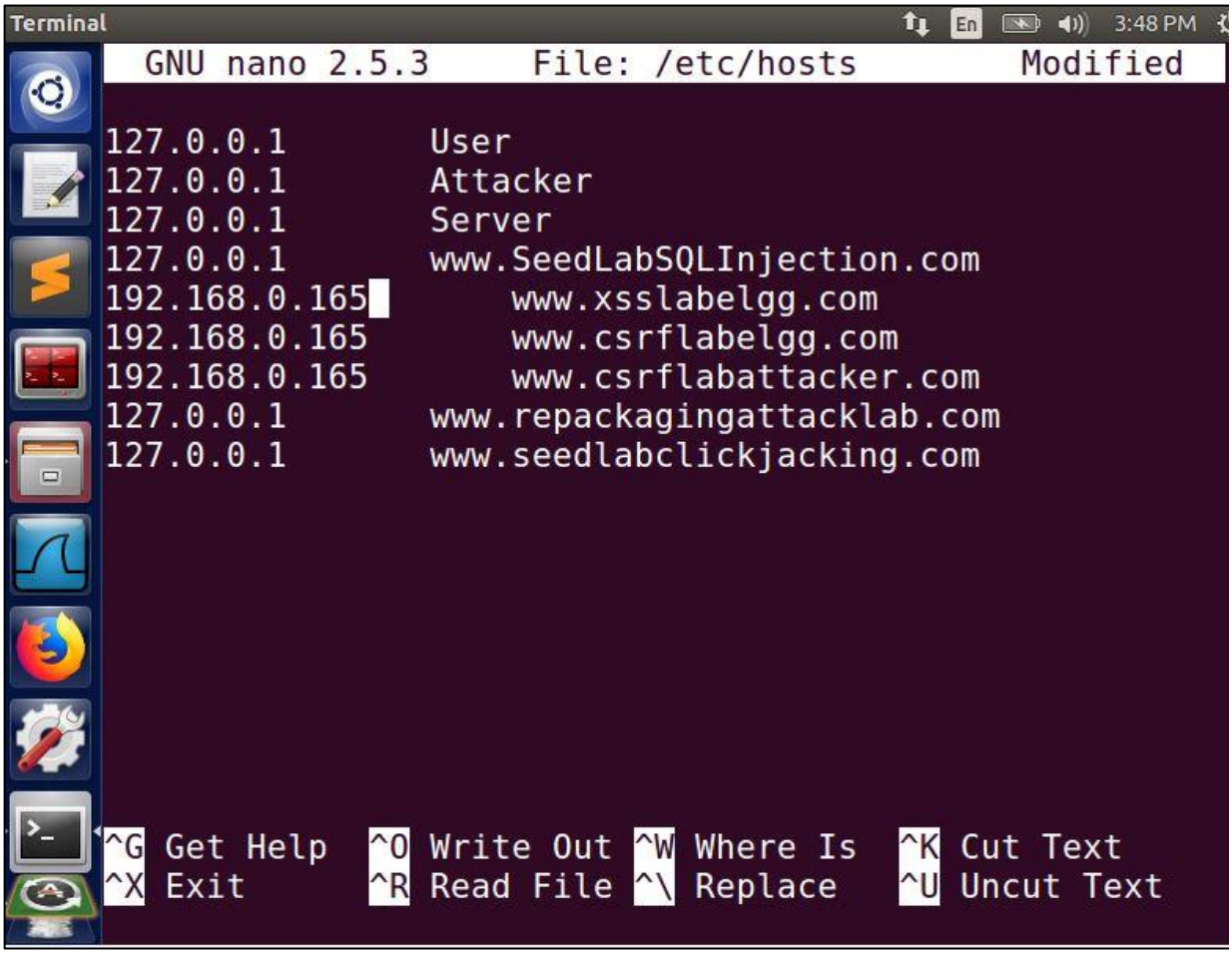

Figure 1: /etc/hosts

4. Apache configuration: Restart apache

Lab Tasks:

NOTE: In this lab, we need to construct HTTP requests. To figure out what an acceptable HTTP request in Elgg looks like, we need to be able to capture and analyze HTTP requests. We can use a Firefox add-on called "HTTP Header Live" for this purpose. Before you start working on this lab, you should get familiar with this tool. Instructions on how to use the HTTP Header Live tool is given in Lab 1.

Task 1: Posting a Malicious Message to Display an Alert Window

The objective of this task is to embed a JavaScript program in your Elgg profile, so when another user visits your profile, the embedded JavaScript code will execute to display an alert widow. The execution result is shown in Figure 2. After login to XSS site as Boby, modify Boby's profile and add an alert script (shown below) in the Brief Description:

```
<script>alert('XSS');</script>
```

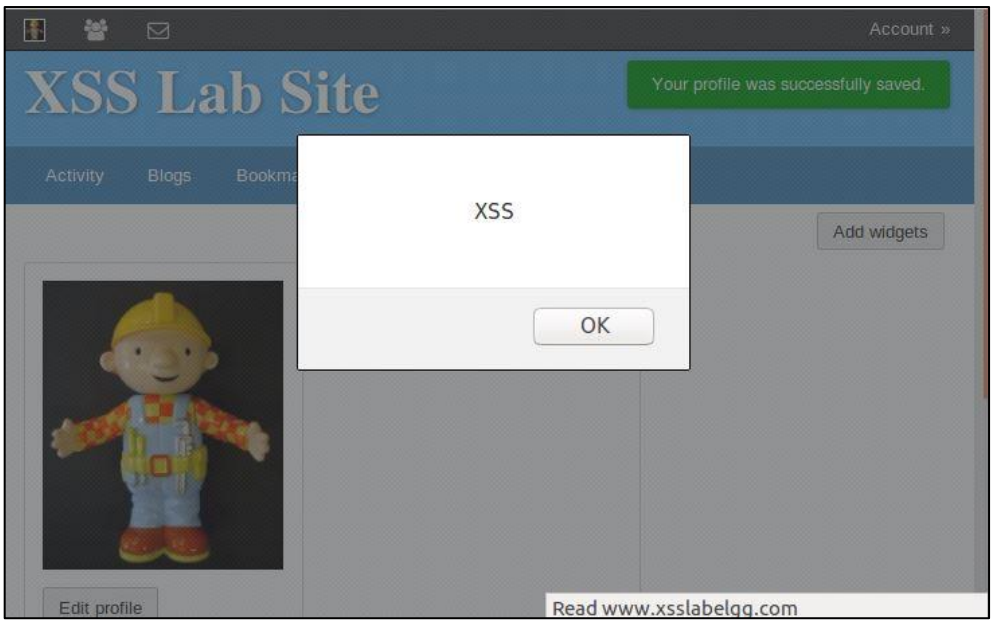

Figure 2: XSS alert

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

The objective of this task is to embed a JavaScript program in your Elgg profile to get other users' cookie information. When another user visits your profile, the user's cookies will be displayed. We login as any one of the Elgg users, modify the user's profile and add an alert script `<script>alert(document.cookie);</script>` in the Brief Description.

The execution result is shown in Figure 3.

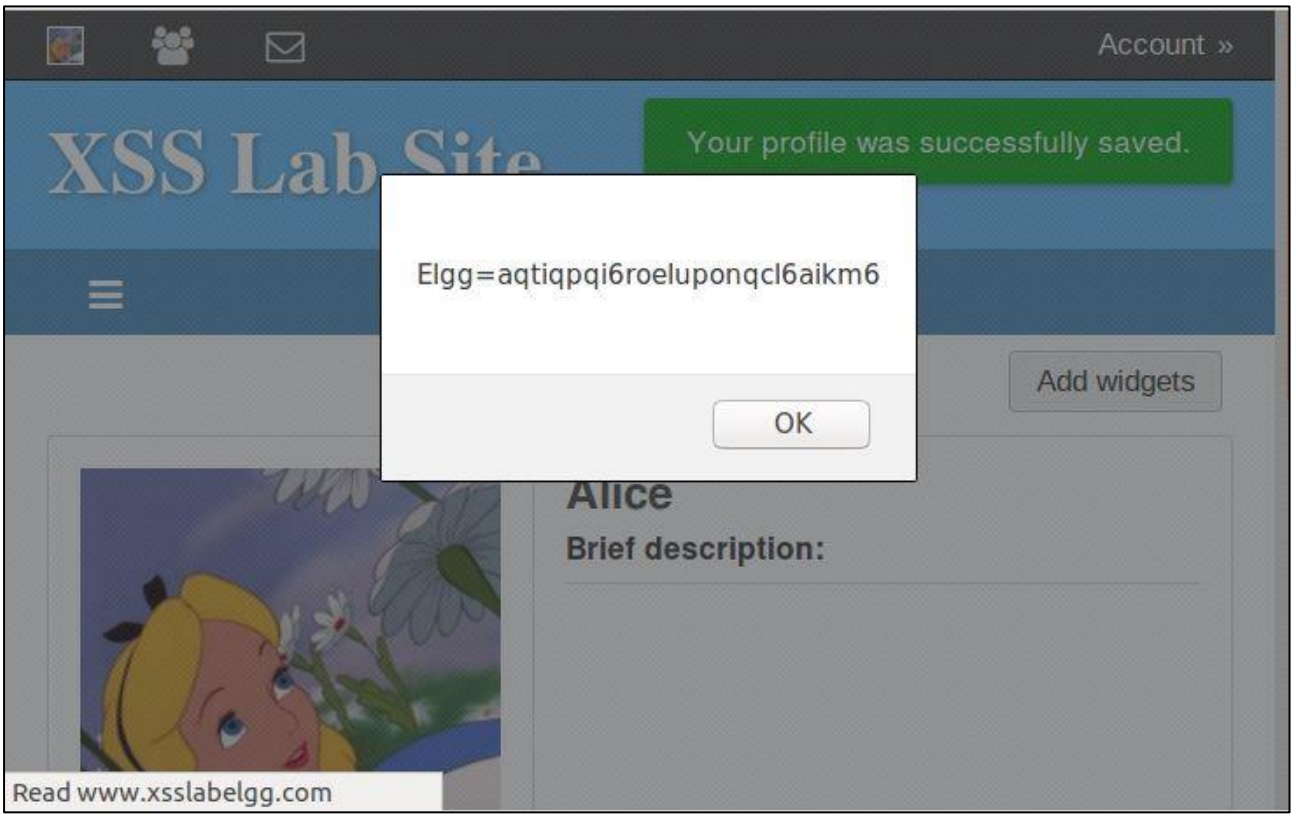

Figure 3: Get cookies

Task 3: Stealing Cookies from the Victim's Machine

In this task, we will steal cookies from a victim's machine. We embed a JavaScript program in your Elgg profile in the Brief Description as shown in Figure 4.

Figure 4: Steal cookies

When the JavaScript code executes, we can cache the cookies in the terminal window as shown in Figure 5.

We can listen to the connection on the specified port and print out whatever is transmitted to that port by typing `nc -l 5555 -v` in the terminal.

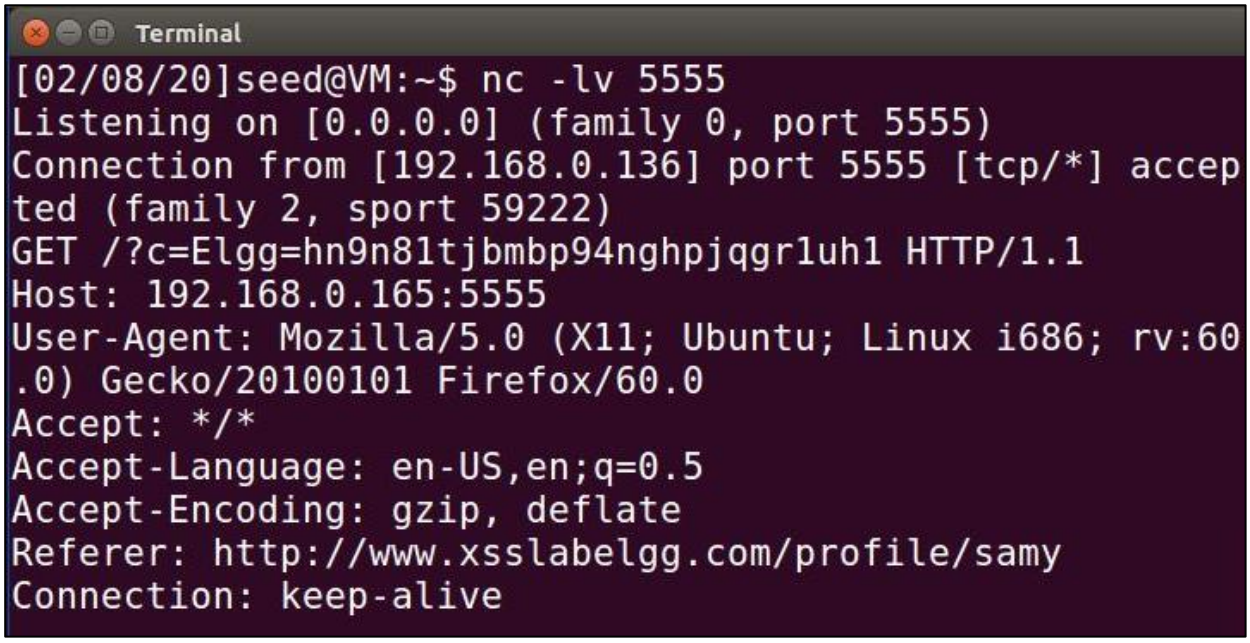

Figure 5: Listening package on port 5555

Task 4: Becoming the Victim's Friend

In this and next task, we will write an XSS worm that adds Samy as a friend to any other user that visits Samy's page. We have to inject code (the worm) to Samy's profile. When other people visit Samy's profile, they will add Samy to their friend list automatically by executing the injected code in Samy's profile. The code (worm) can be found in the Seed Lab Web_XSS_Elgg. In this task, Samy will be an attacker and create various techniques that can be used for this attack.

Figure 6 shows the guid number and token when we want to add Samy to our friend list. In the HTTP Header Live, we can see the detail of information for adding Samy to the friend list.

Figure 6: HTTP POST request

Now, we need to write a JavaScript program to send out the same HTTP request. We inject the code in Samy's profile. Figure 7 shows the code added in the About Me field by clicking "Edit HTML".

Figure 7: Injecting code

As shown in Figure 8, before we inject the code in Samy's profile, Samy is not Alice's friend yet.

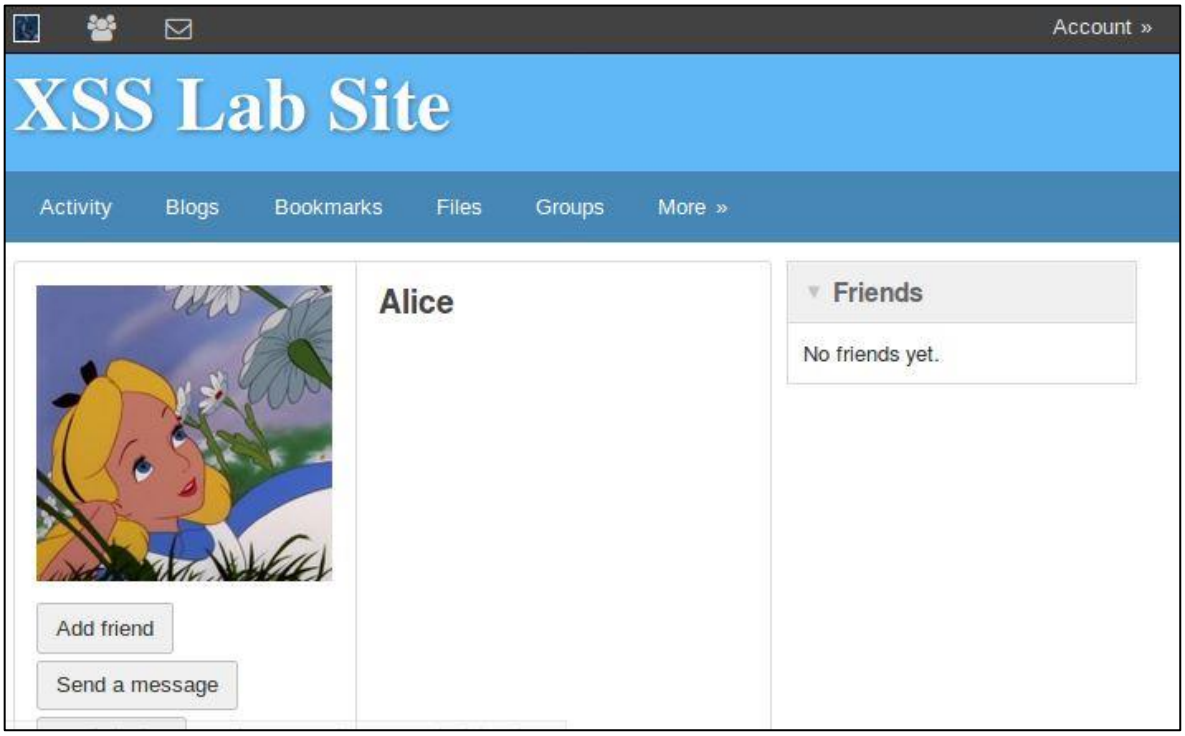

Figure 8: Alice's profile

After we have added the code, Samy's profile should look like what is shown in Figure 9.

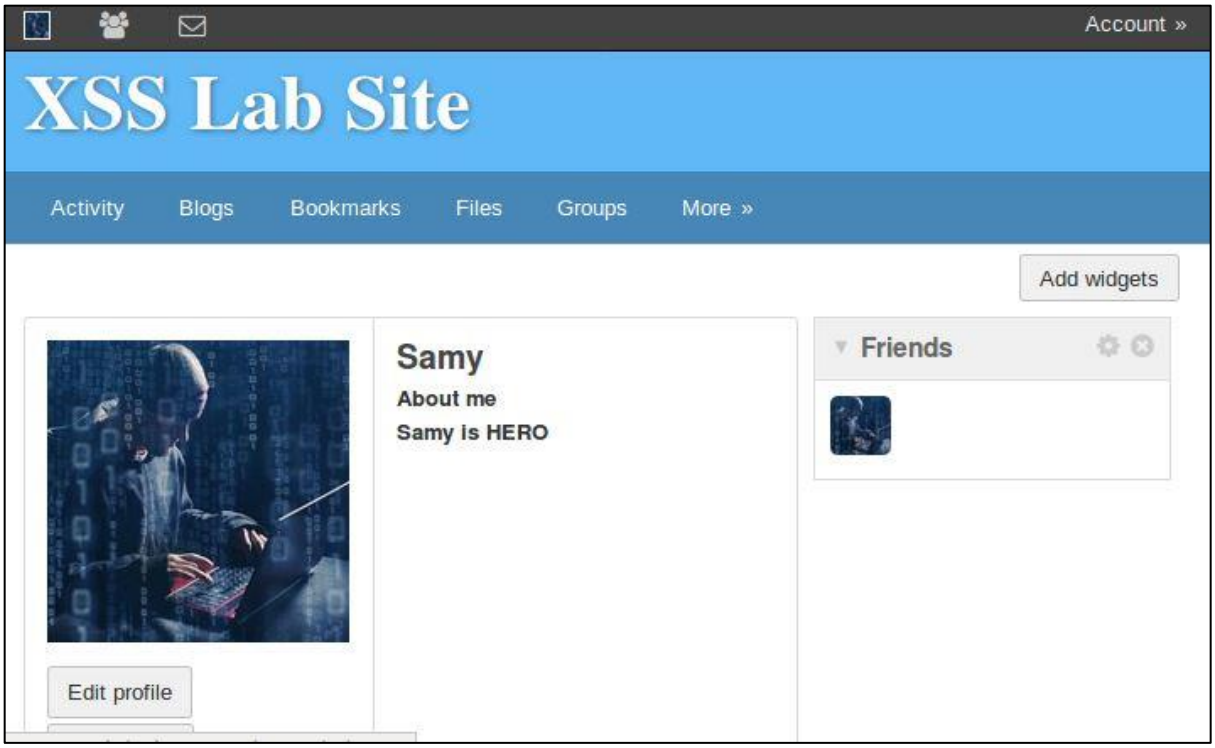

Figure 9: Samy's profile

When Alice visits Samy's profile, Alice will add Samy to the friend list (by the code injected in Samy's profile) as shown in Figure 10 and Figure 11.

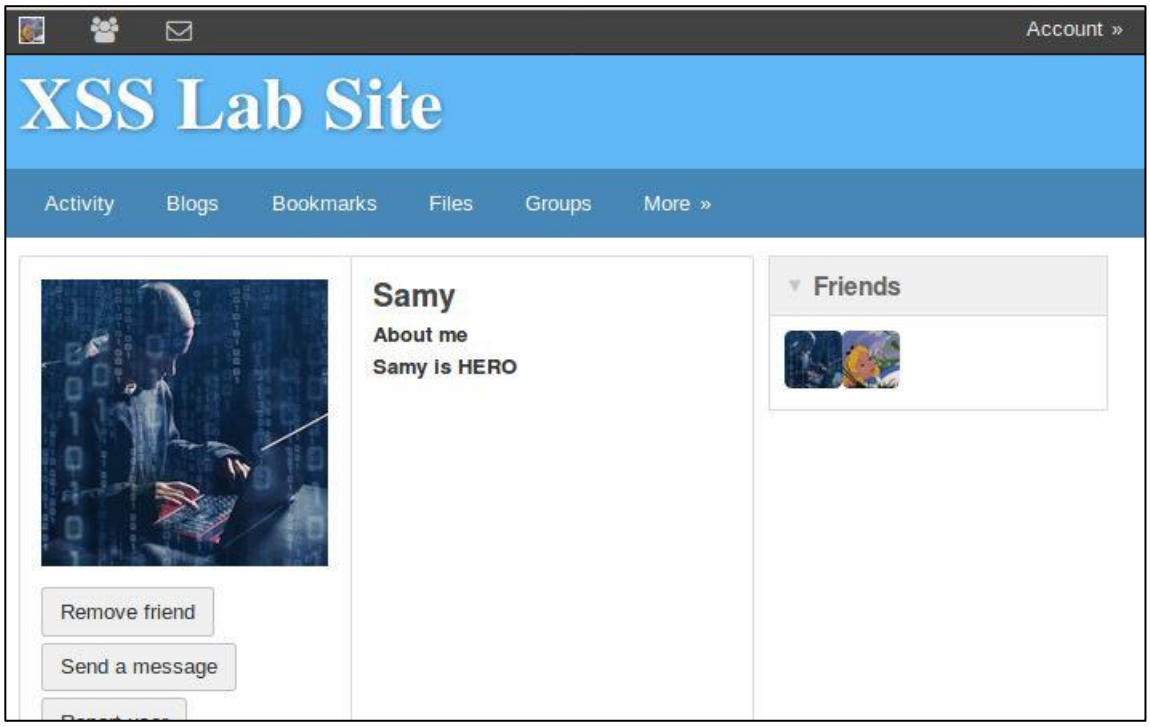

Figure 10: Alice visits Samy's profile

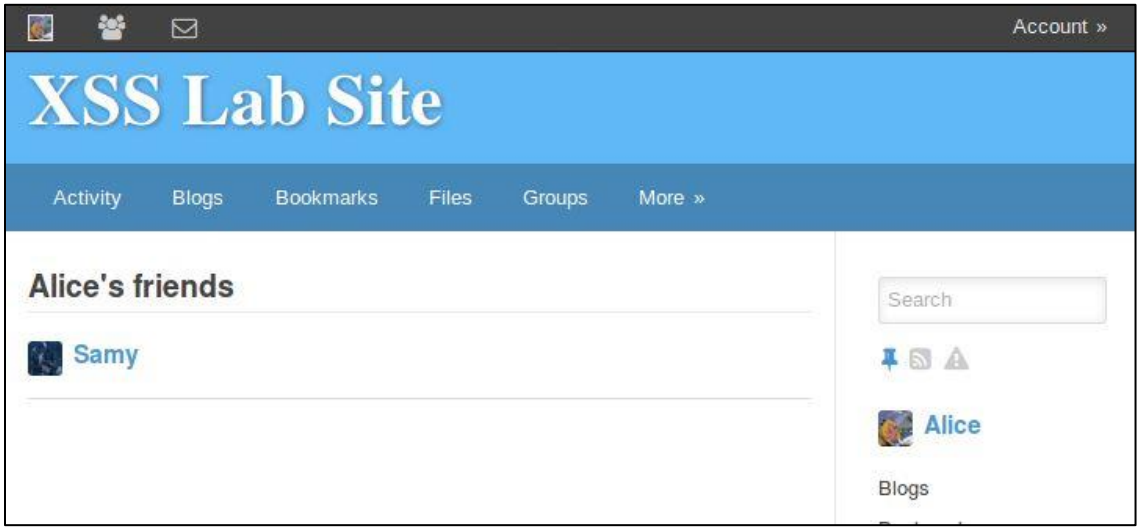

Figure 11: Alice added Samy as friend

Task 5: Modifying the Victim's Profile

In this task, we need to modify the victim's profile by adding a malicious JavaScript program that forges HTTP requests directly from the victim's browser. We will write a JavaScript program to send out the HTTP request to the user and modifying a victim's profile. As the result, when the victim visits the attacker's profile, the victim's profile modified by the attacker automatically. Figure 26 shown the JavaScript in the attacker's profile. Please preform the attack and answer the additional questions in the instruction.

To figure out how Samy would forge a POST request, we need to investigate how the HTTP request would trigger when we edit the profile. Figure 12 below shows the HTTP request.

Figure 12: HTTP POST request

Figure 13 shows the code added in the About Me by clicking “Edit HTML”.

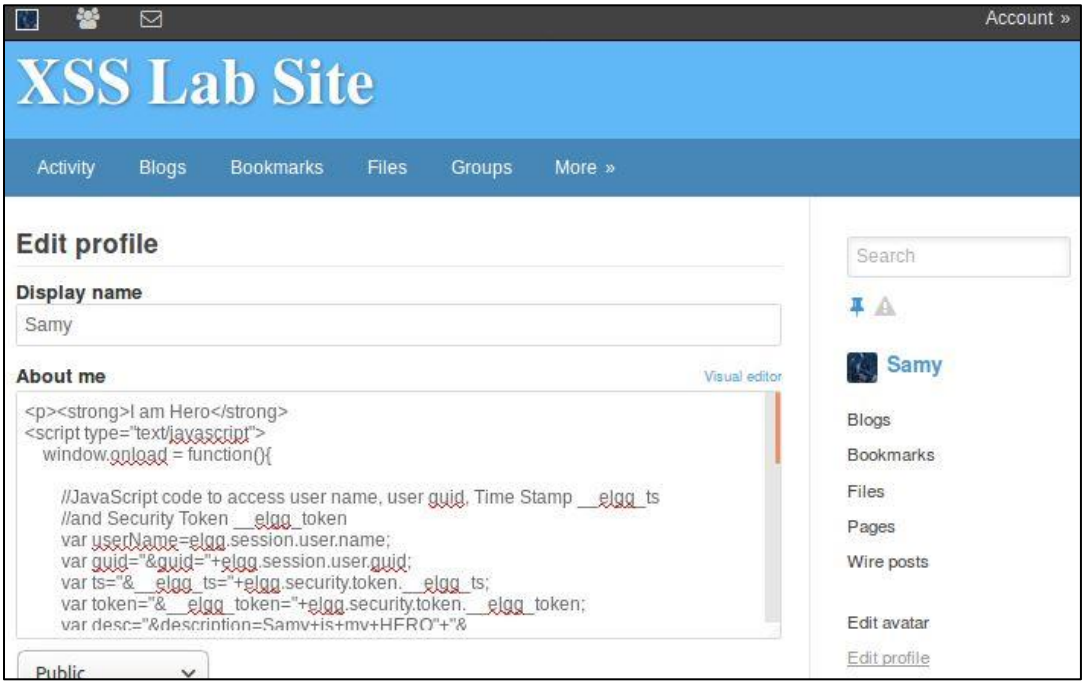

Figure 13: Injecting code in Samy’s profile

Logging in as Alice, we can see Alice’s profile has not been modified yet (as shown in Figure 14).

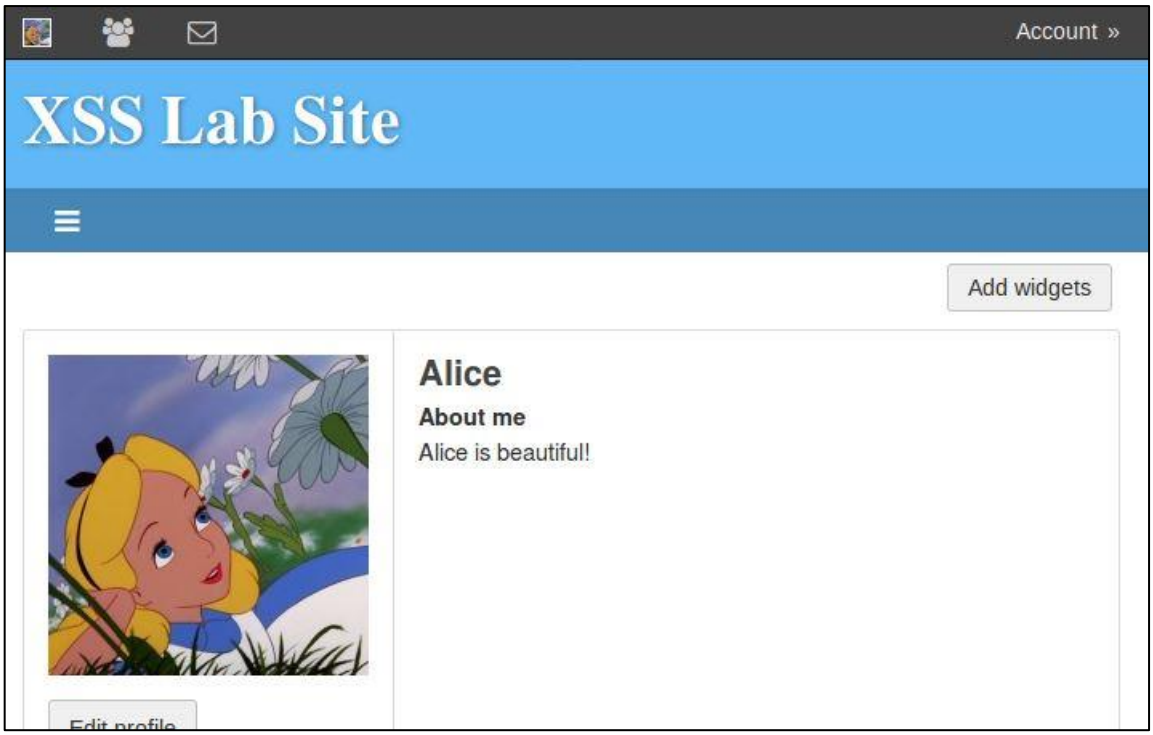

Figure 14: Alice’s profile

When Alice visits Samy's profile, we can see Samy's profile (as shown in Figure 15).

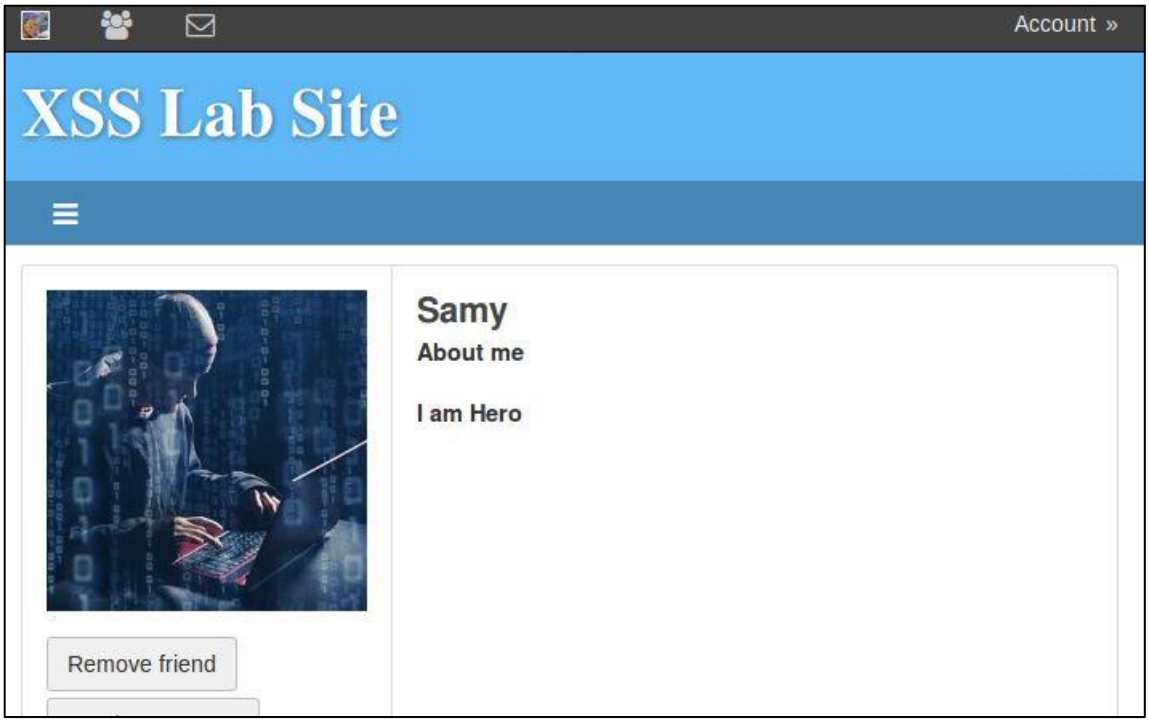

Figure 15: Alice visits Samy's profile

After visiting Samy's profile, Alice's profile will be modified by Samy automatically. Figure 16 shows the modified Alice's profile.

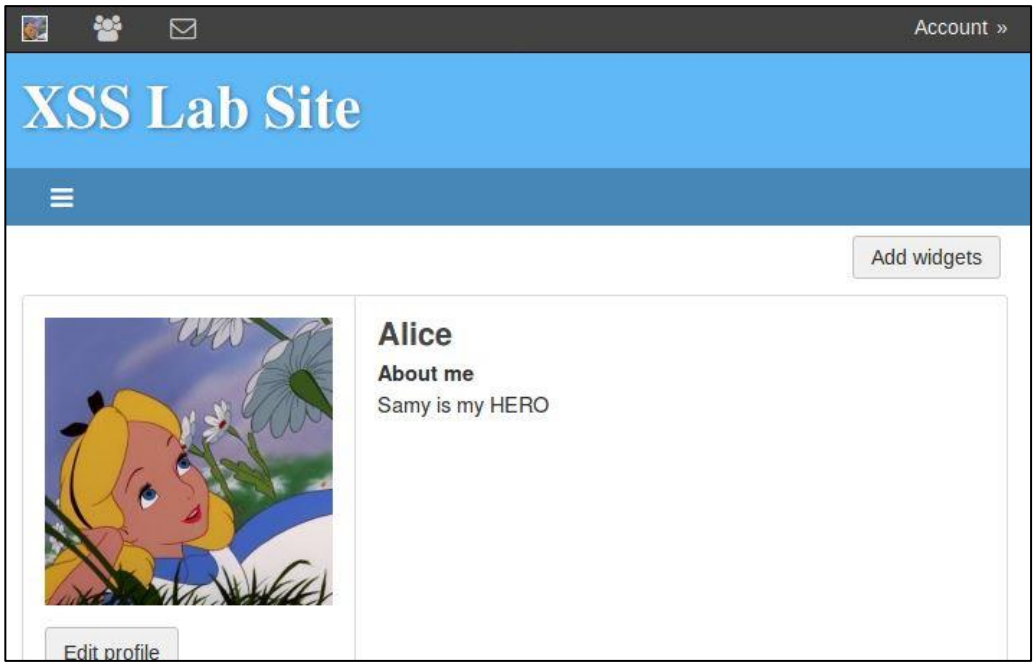

Figure 16: After Alice visited Samy's profile

We can observe HTTP request when Alice visits Samy's profile. Figure 17 shows that Samy has successfully modified Alice's profile.

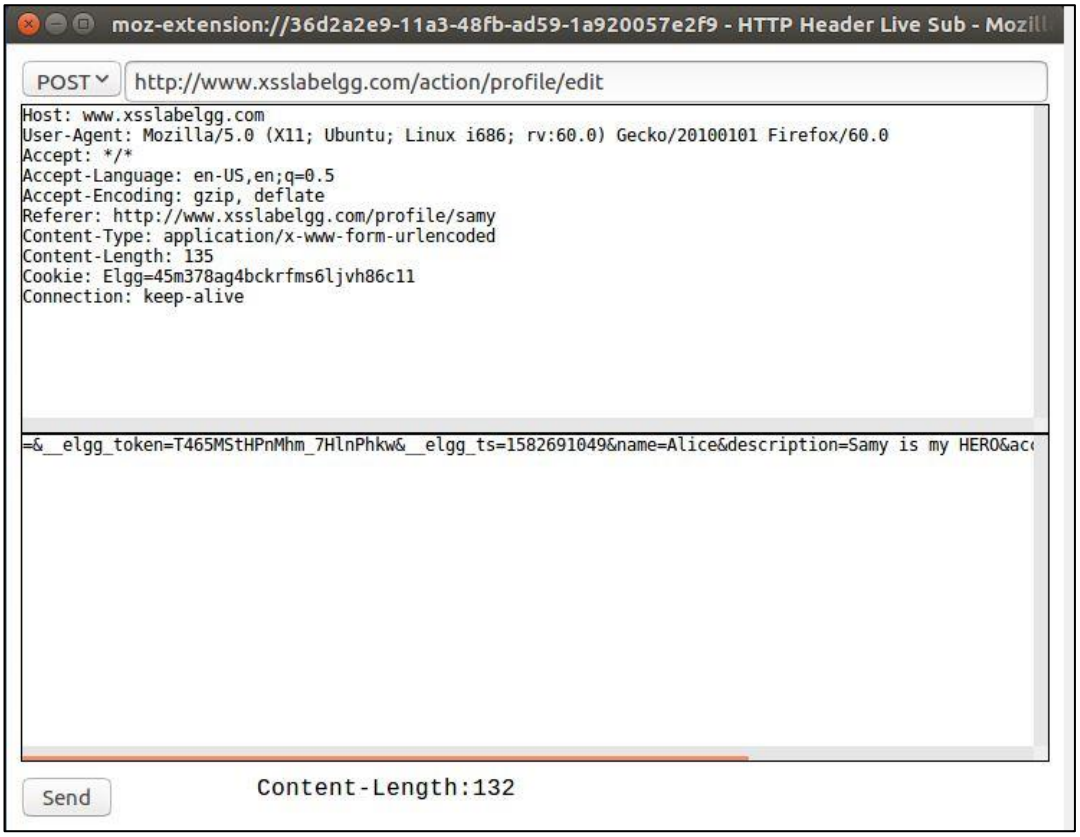

Figure 17: HTTP POST request

Task 6: Writing a Self-Propagating XSS Worm

In this task, we will perform a self-propagating worm to modify the user profile and self-propagating itself to other user's profiles. The more users visit a victim's profile, the more attackers will be. First, we have to inject the code(worm) to Samy's profile. When a user visits Samy's profile, the injected code will execute and modify a victim's profile. Second, after the victim visited Samy's profile, the code will retrieve a copy of it from the DOM tree of the webpage. Third, when the other user visits the victim, the self-propagate code will duplicate to the other user and so on. Please preform the attack and answer the additional questions in the instruction.

Figure 18 shows Alice's profile before visiting Samy. Figure 19 shows the JavaScript self-propagating program in Samy's profile. Figure 20 shows Samy's profile after the self-propagate program injected. Figure 21,22, and 23 shown the self-propagating program injection after the other users visit Boby's profile.

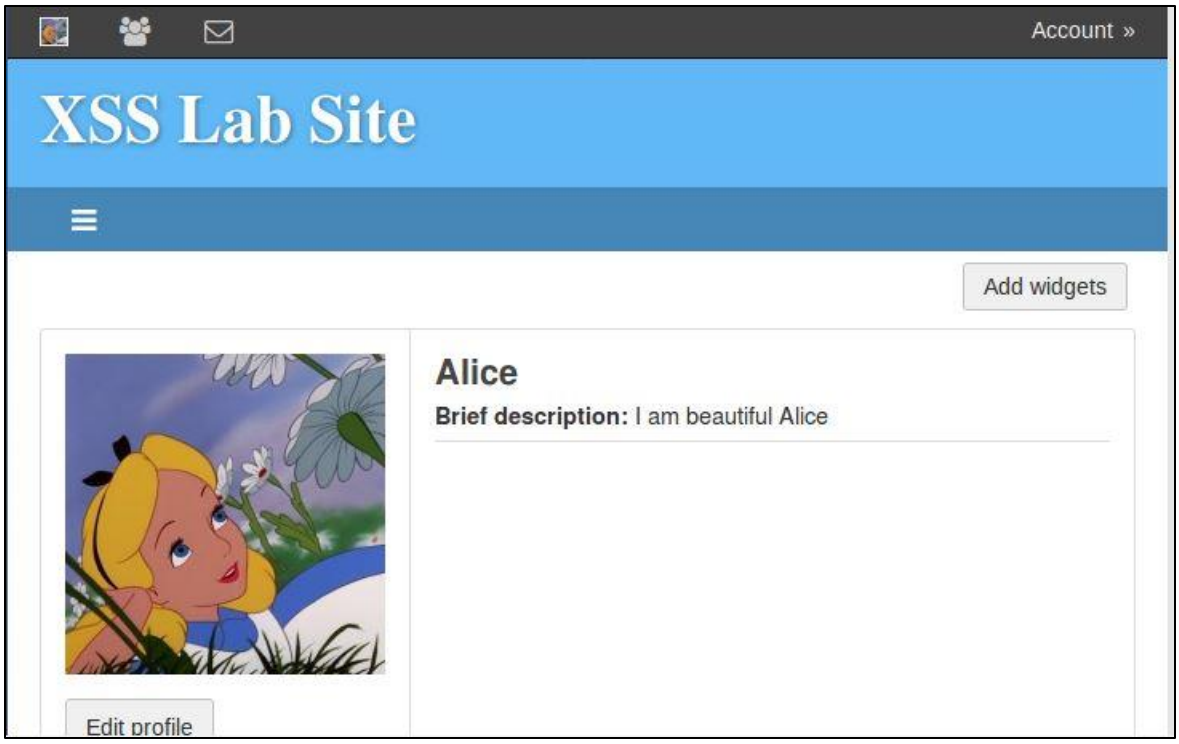

Figure 18: Alice's profile.

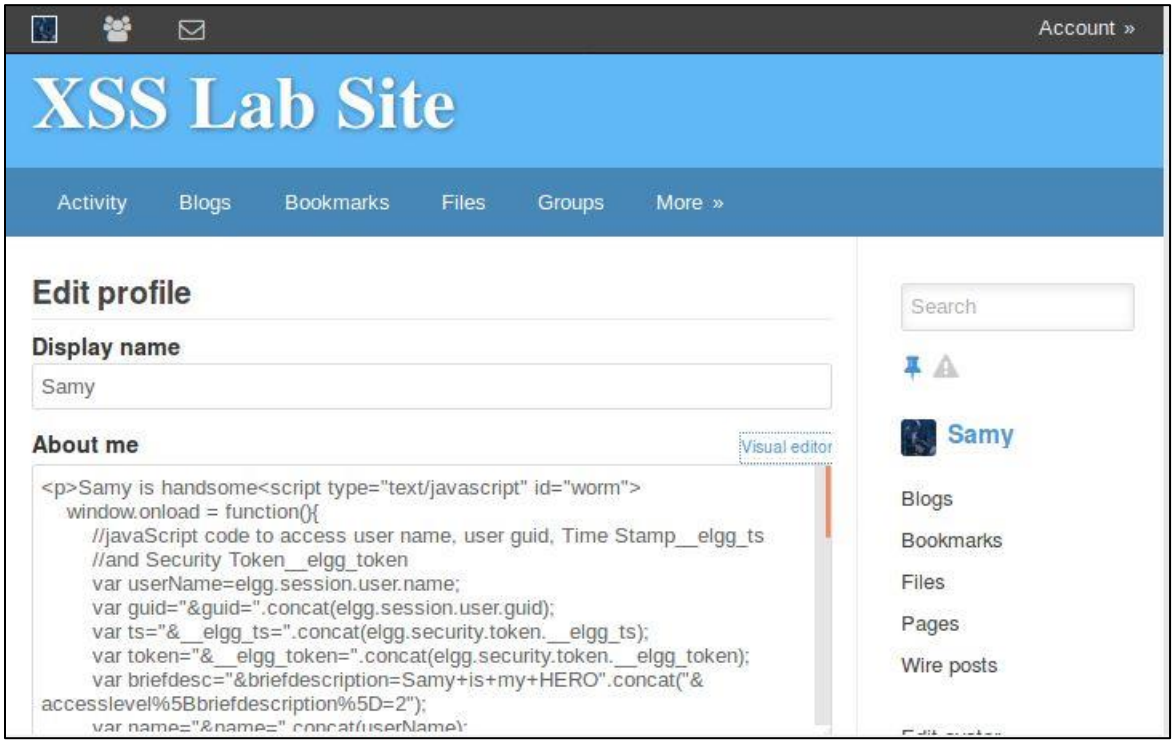

Figure 19 The JavaScript self-propagating program in Samy's profile

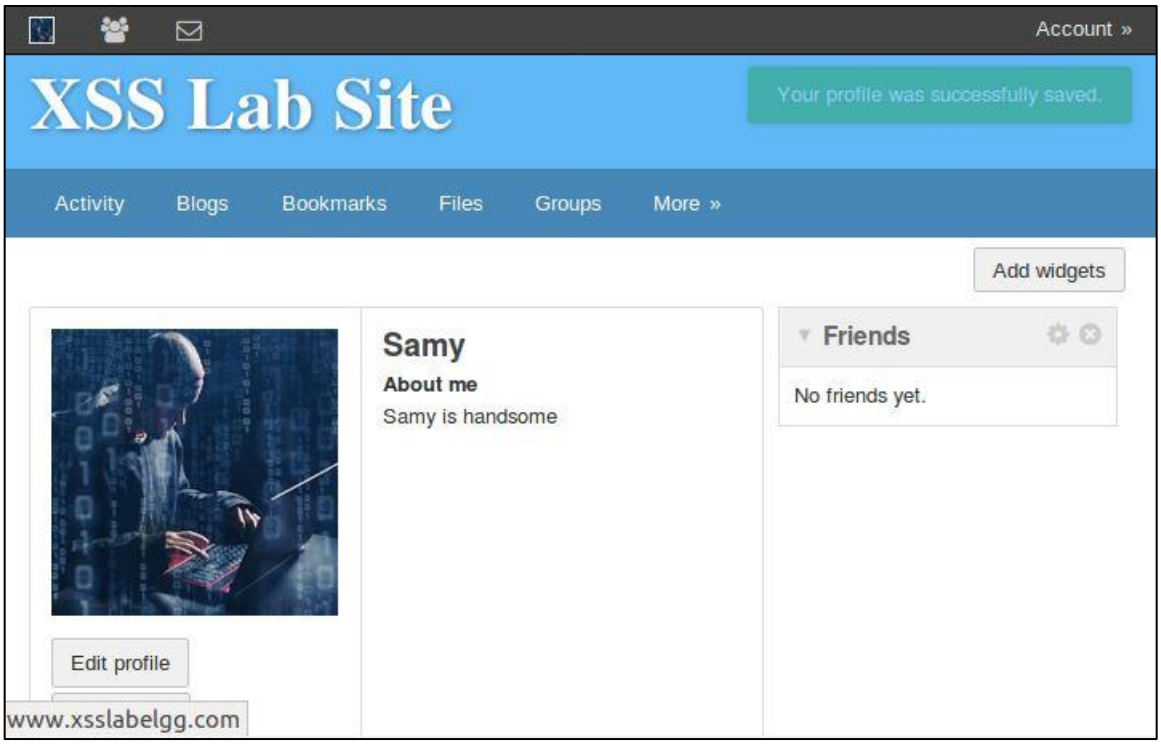

Figure 20: Samy's profile after the JavaScript self-propagating program written.

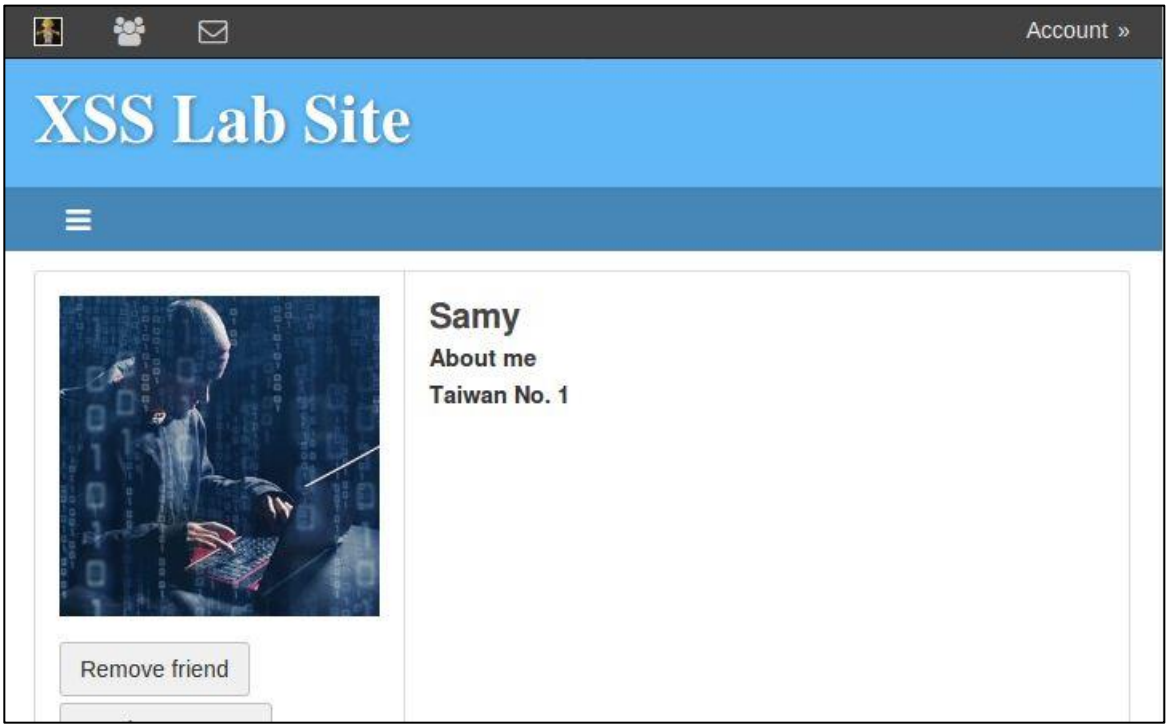

Figure 21: Boby visited Samy's profile

Figure 22: Boby's profile after visiting Samy's profile.

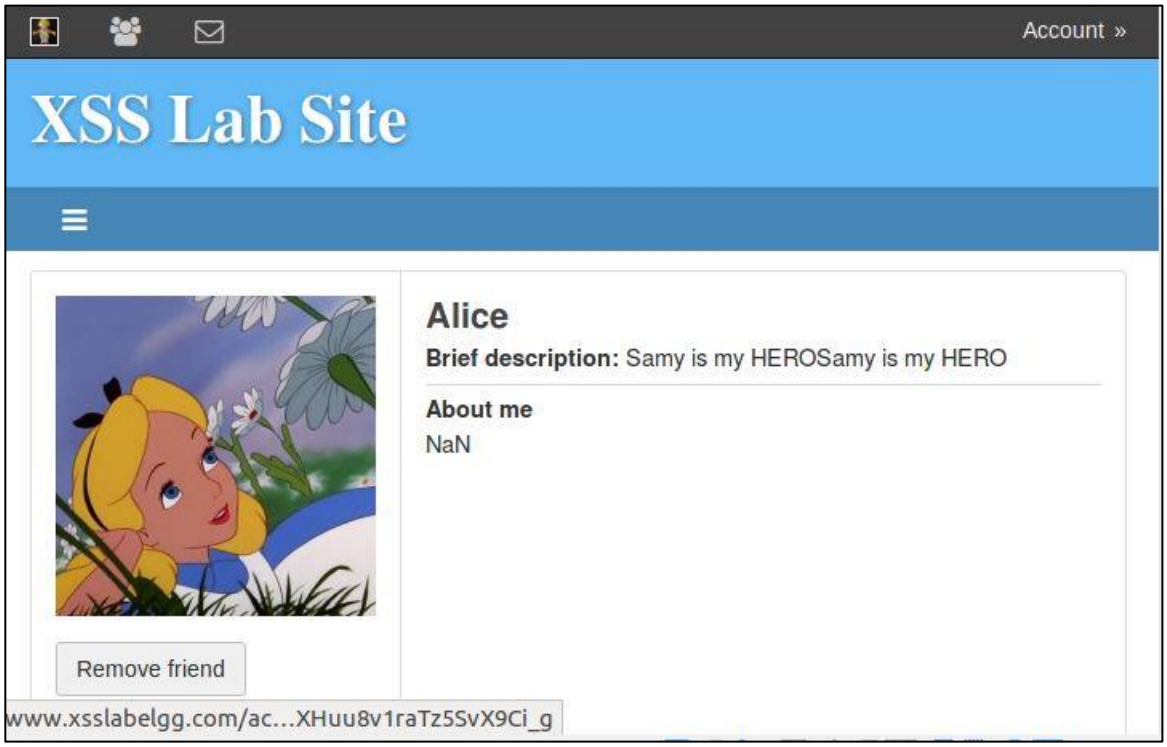

Figure 23: Alice's profile after visiting Boby's profile.

Figure 24 shows Boby copied the JavaScript self-propagating program after visiting Samy's profile.

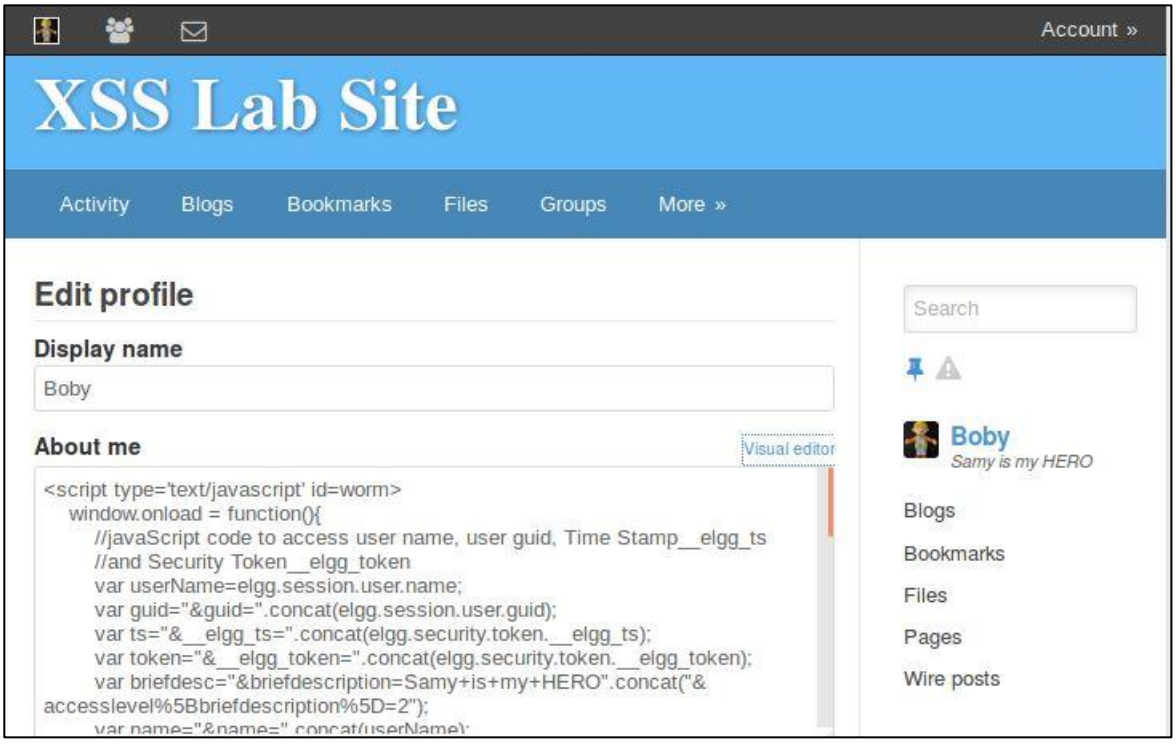

Figure 24: Edit Boby's profile.

Task7: Countermeasures

In the previous attack, we learned how to perform the attack to infect other users. In this task, we will learn how to turn on countermeasurement for preventing the attack.

First, turn on and off HTMLawed in the Elgg website by logging in as admin. After logging in as the admin, click the “account” tab on the right corner to bring up the administration page (Figure 25), and then click plugins on the right bar. Activate HTMLawed and describe your observation (Figure 26).

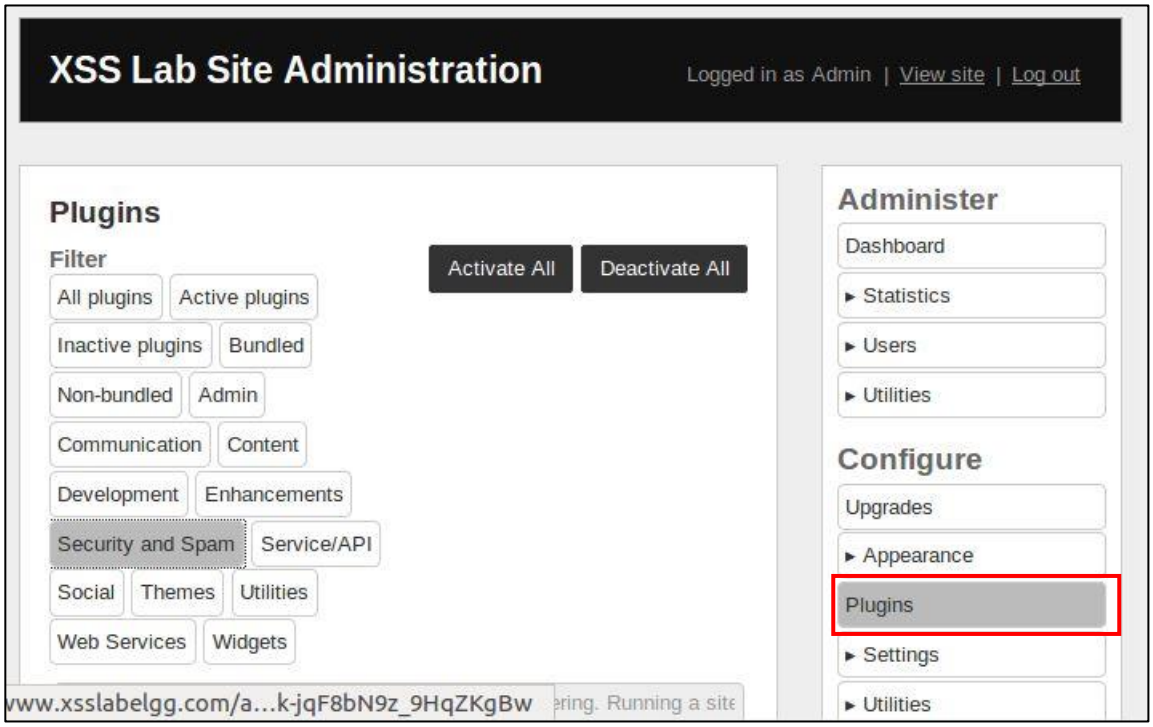

Figure 25: Go to Plugins

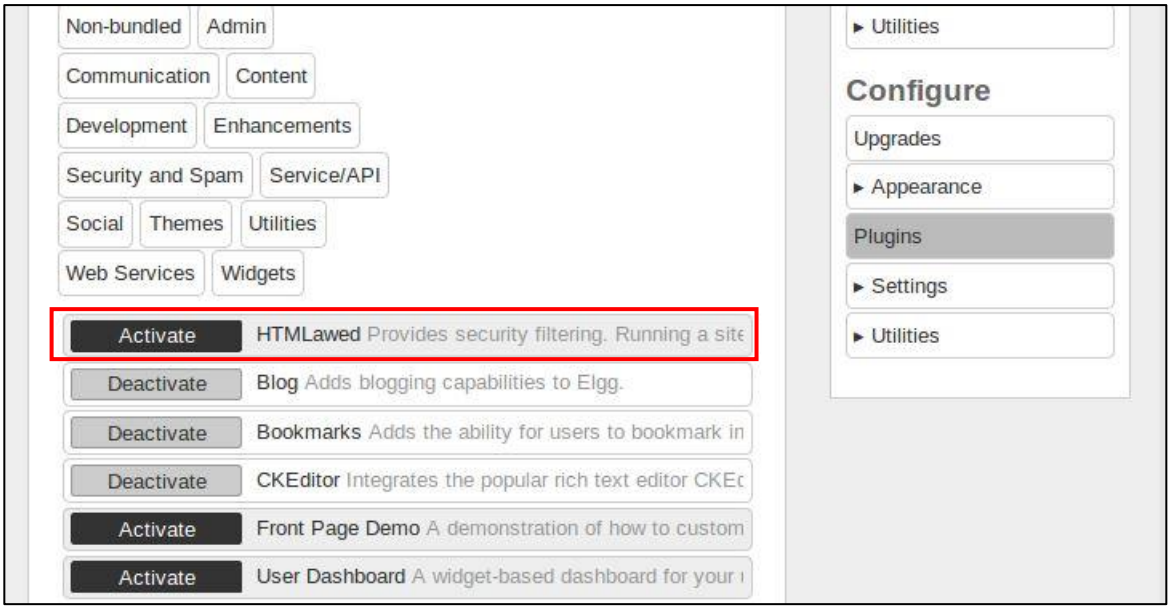

Figure 26: Activate HTMLawed

Second, turn on the encoding special characters in user input by opening the files “text.php, url.php, dropdown.php and email.php” in the folder /var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output. Uncomment the corresponding "htmlspecialchars" function calls in each file and do not change any code. Please preform the countermeasurement and answer the additional questions in the instruction. Additional information and tutorial can be found in the textbook and Seed Lab. Figure 27 shows the "htmlspecialchars" functions that should be uncommented.

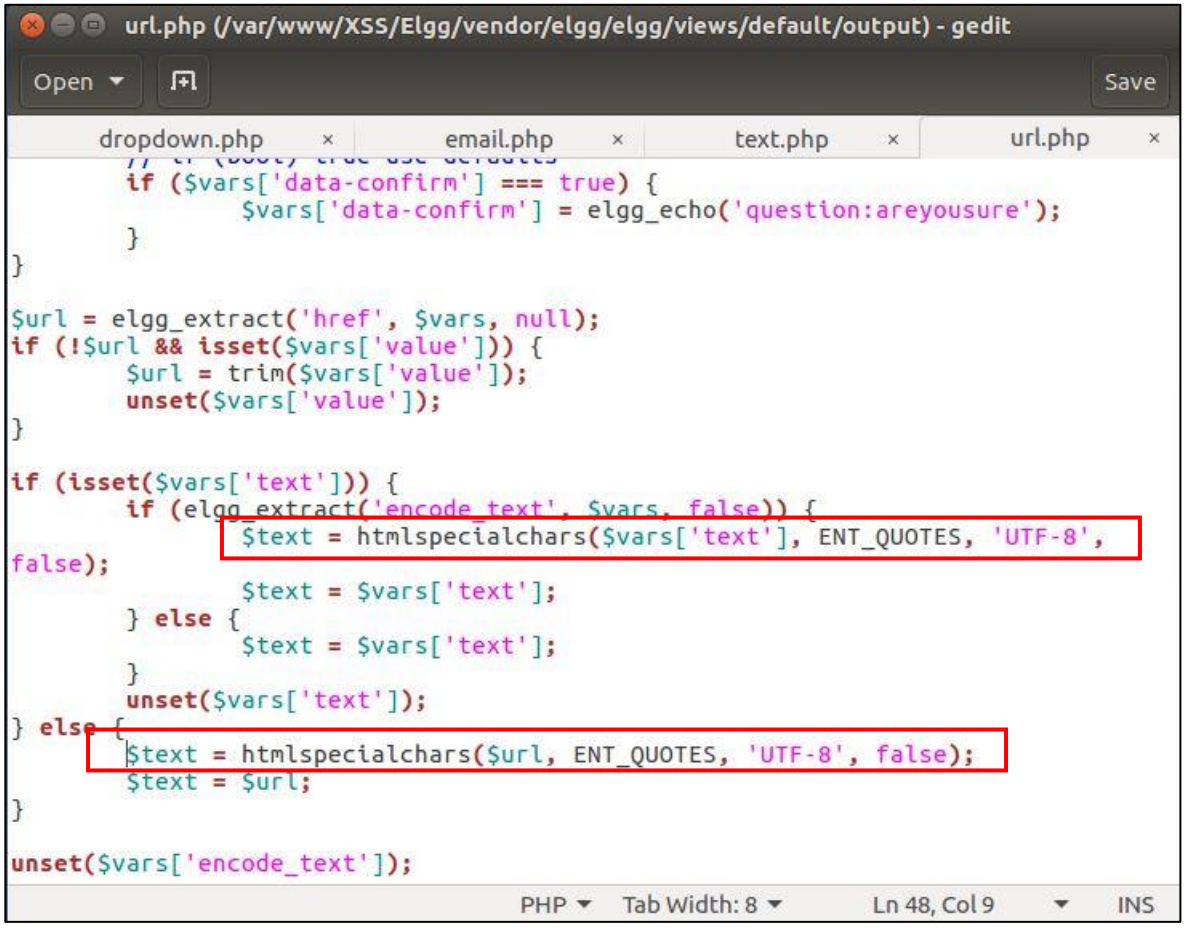

Figure 27: Uncomment the "htmlspecialchars" function

To test the countermeasurement, you can test the attack we used in task 1. When you turn on the countermeasurement, the injected attack code should convert special characters to HTML entities.