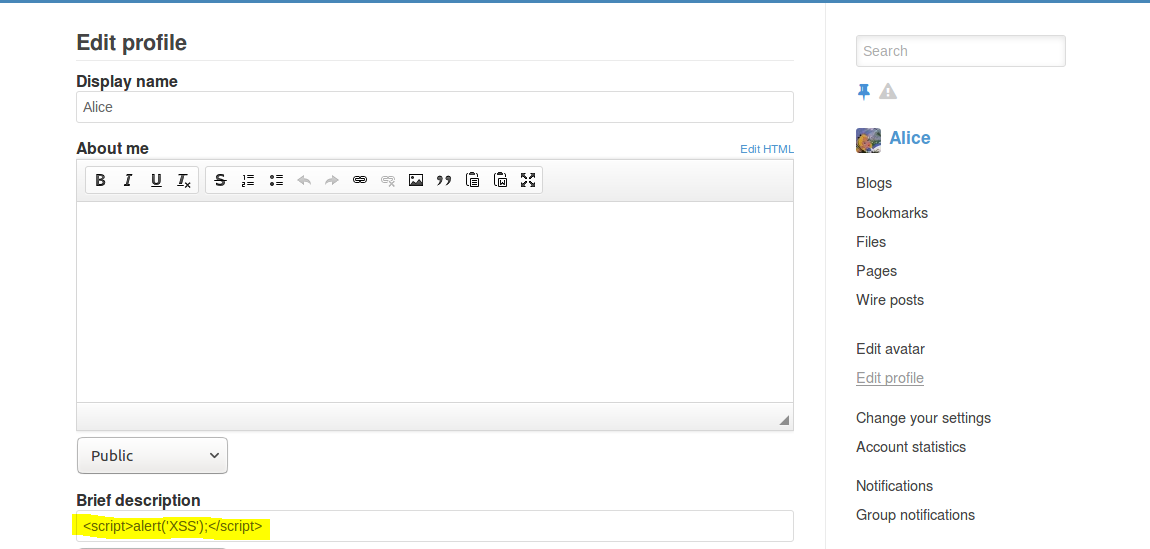
**Cross Site Scripting Attack Lab**

**Karan Amrutesh**

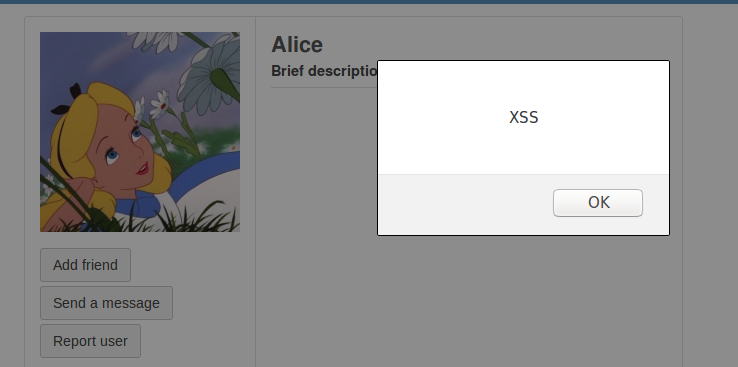
**Task 1: Posting a Malicious Message to Display an Alert Window:**

Injecting a script code in the brief description field of Samy’s profile:



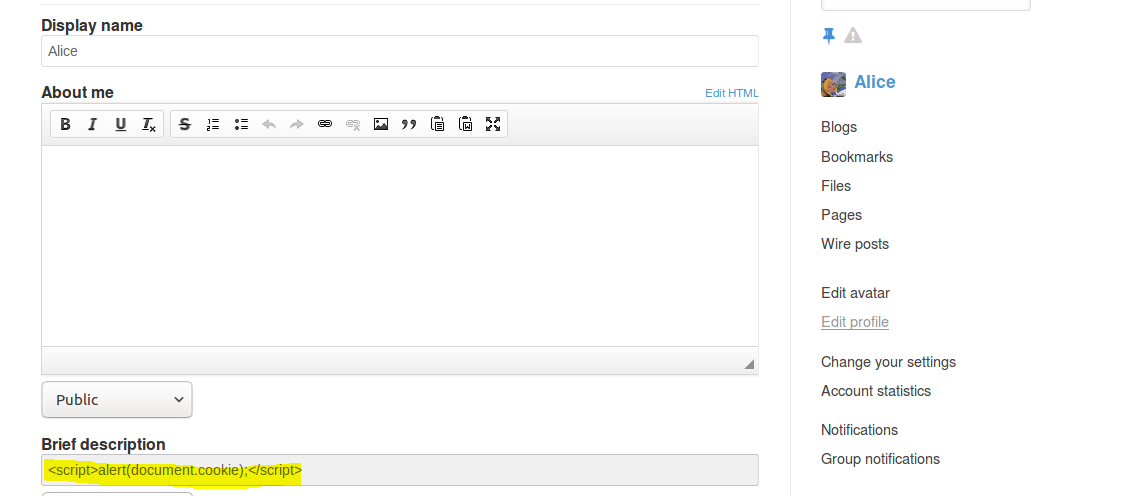
We can see that when any user(including Samy) visits Samy’s profile, the java script code is executed and an alert message is displayed:

The alert window is displayed wherever the brief description field in present, including the member’s page:

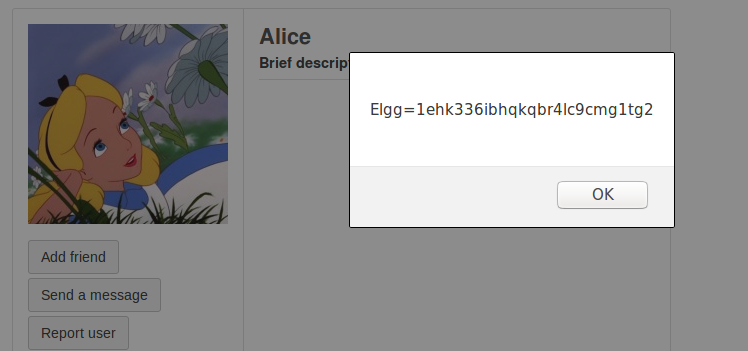


**Task 2: Posting a Malicious Message to Display Cookies**

We add the malicious script code to display the cookies in the brief description field of Samy’s profile.

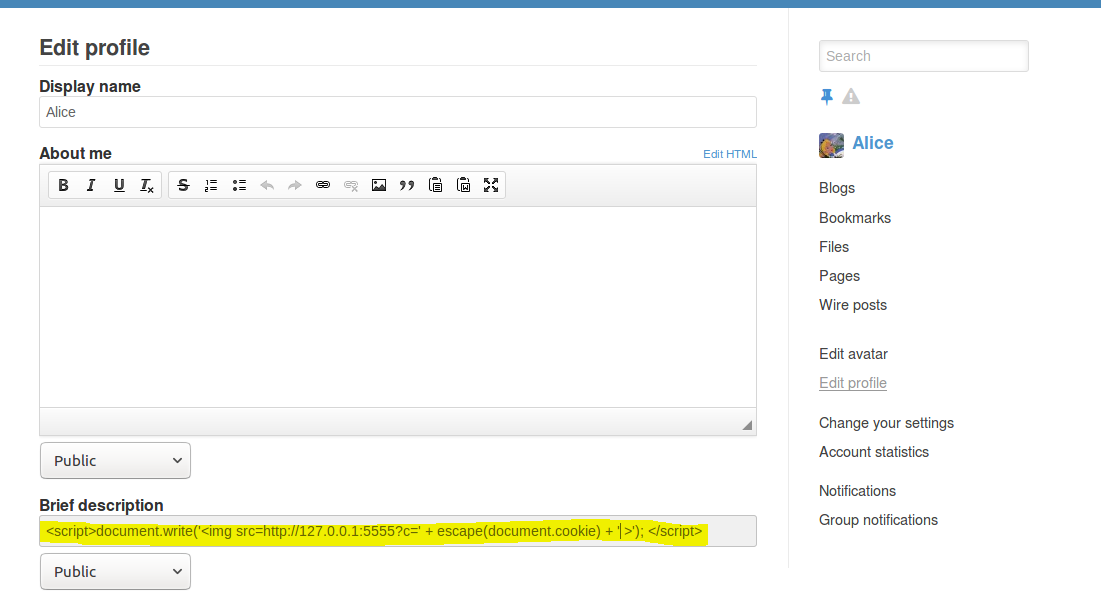


So when any user visits Samy’s profile, their cookie is displayed:

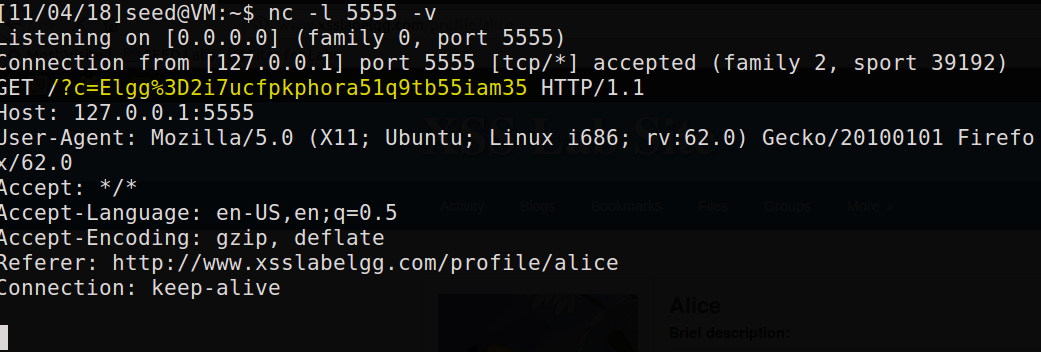


**Task 3: Stealing Cookies from the Victim’s Machine**

We add the malicious script code to steal the victim’s cookies in the brief description field of Samy’s profile.

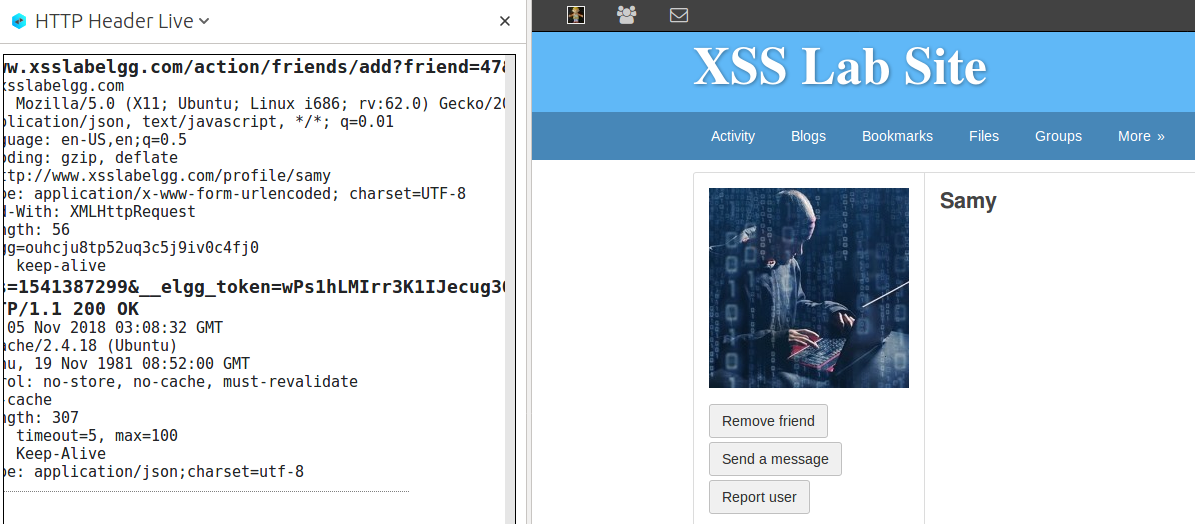


* The img tag will send HTTP GET request to the attacker(specified my IP address) along with the victim’s cookie to port 5555.
* Listening on port 5555 on localhost, we can see that the victim’s cookie is obtained:



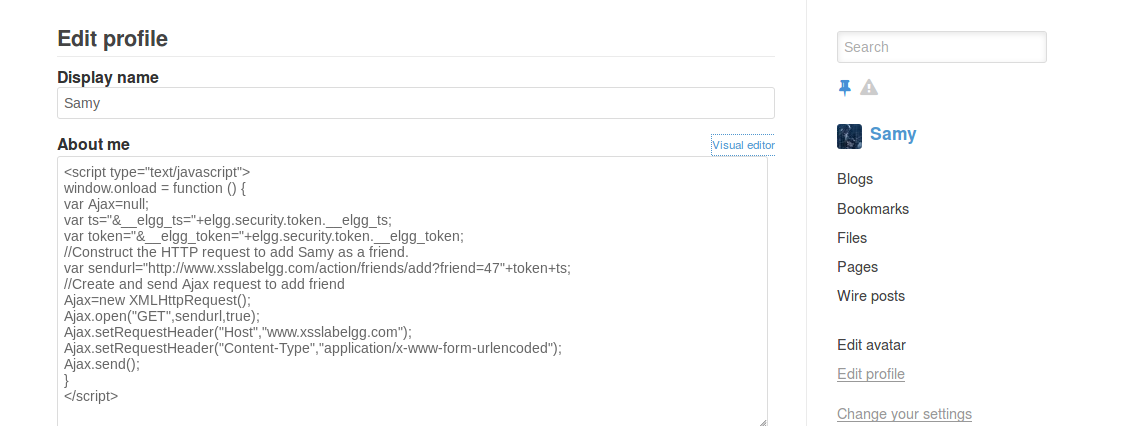
**Task 4: Becoming the Victim’s Friend**

We first visit Samy’s profile from Boby’s account and click on Add Friend to get the URL and Samy’s GUID which is 47.

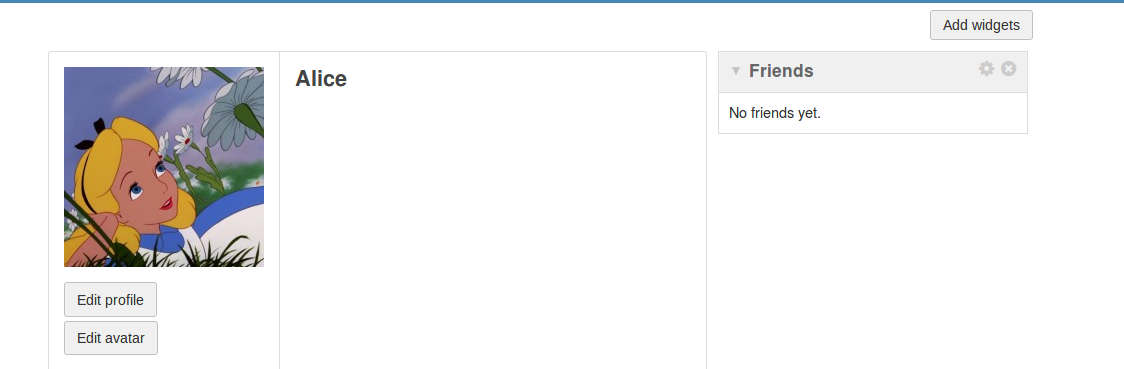


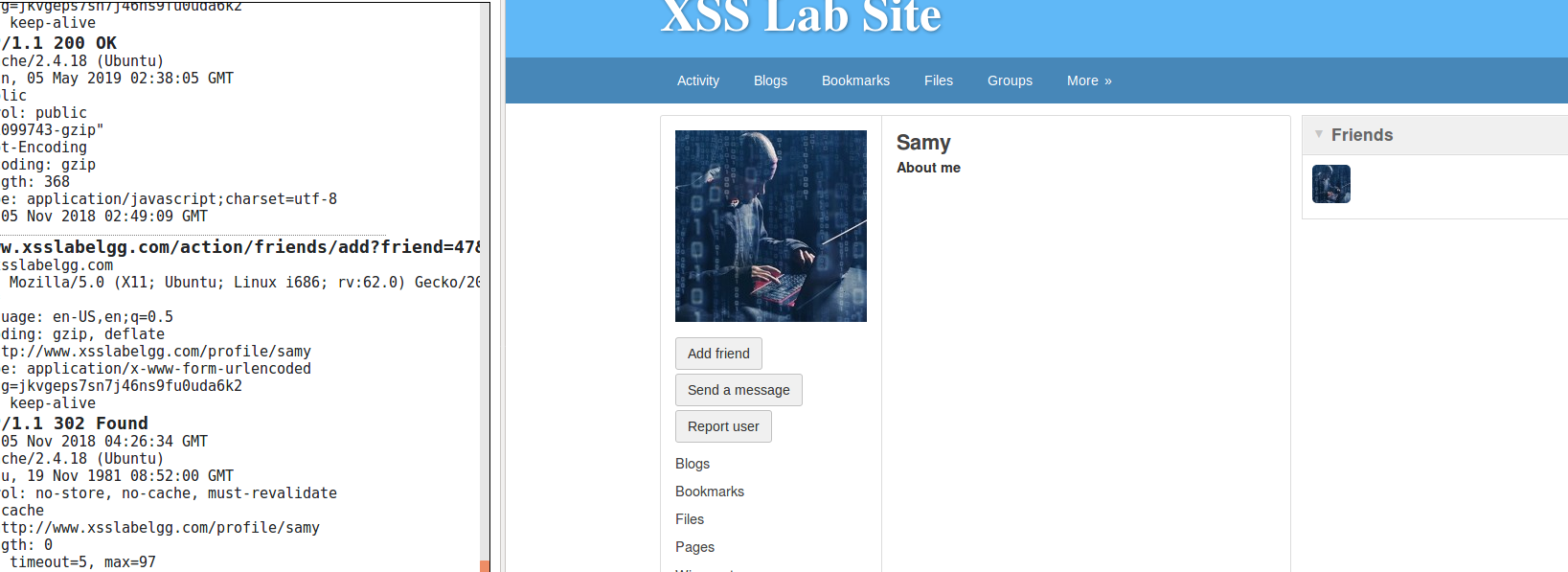
Then we edit Samy’s profile, injecting the script code to add Samy has a friend in his About Me field.

We construct the url by the investigation done above, adding the token and timestamp as well.

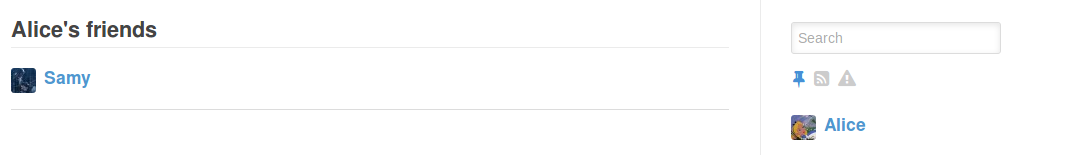


Before the attack, we can see that Alice does not have Samy as her friend:





Then when Alice visit’s Samy’s profile, we can see that the script code is executed and Samy is added Alice’s friend list. It can be verified by viewing the HTTP Header Live.



Please answer the following questions:

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

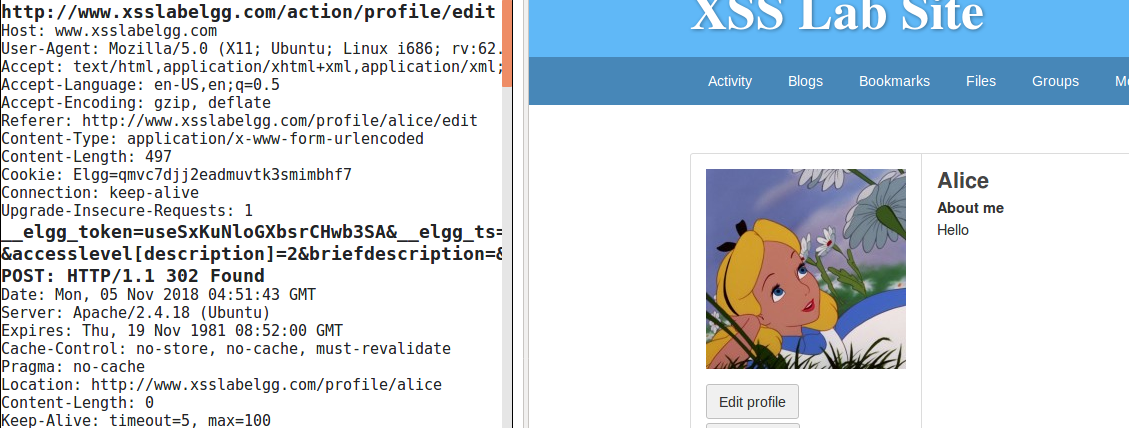
These two lines are needed to get the secret token and the timestamp of when the token was sent. We store these two parameters into variables to make accessing easier.

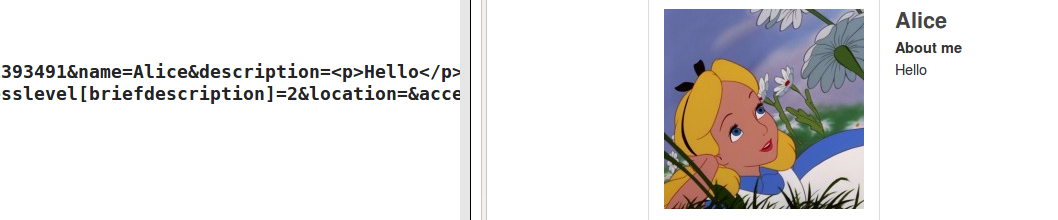
Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?

Yes, the attack can still be launched successfully by using some browser extension to remove the formatting data from the HTTP requests done by the editor.

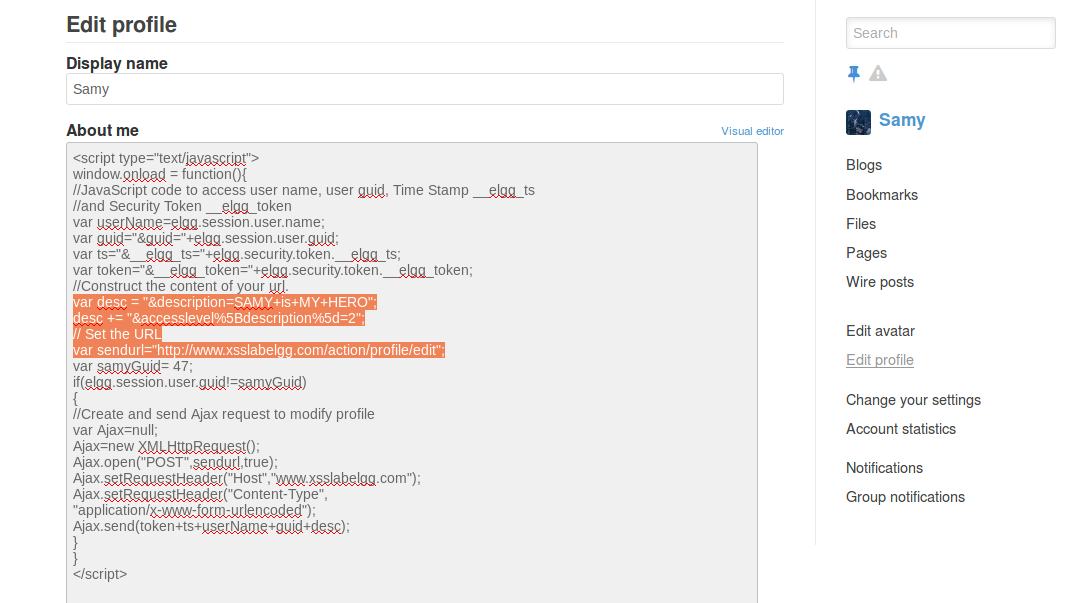
**Task 5: Modifying the Victim’s Profile**

We get the url from editing Alice’s About me field and observing using HTTP Header Live:

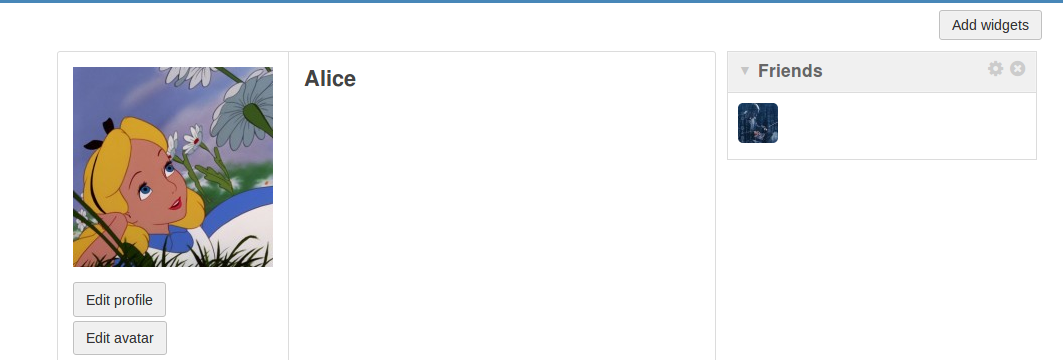




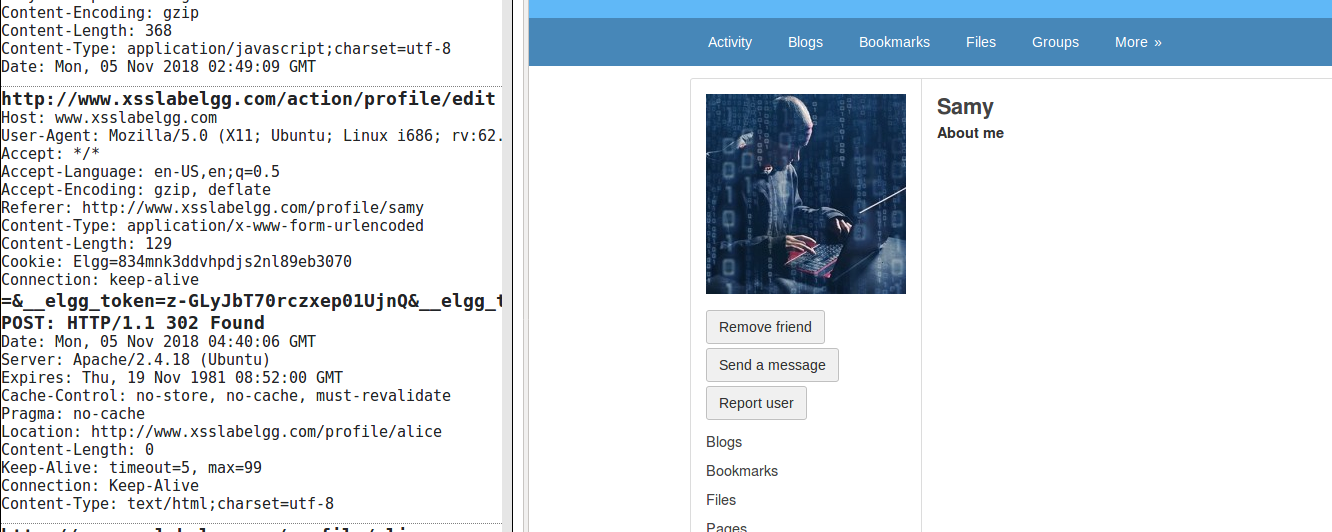
Then we use this URL in the malicious code that we inject in Samy’s profile:

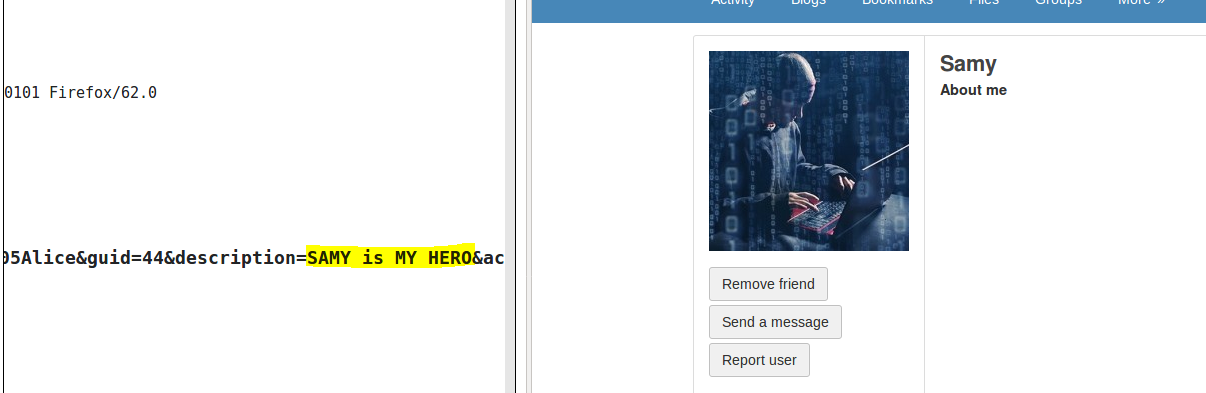


Alice’s profile before the attack:

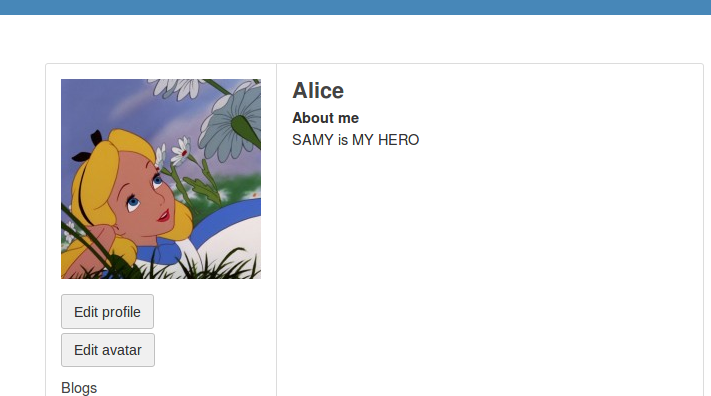


When Alice visits Samy’s profile, we can see that the script put in the About me section of Samy’s profile is executed and a POSt request is sent to modify Alice’s profile:



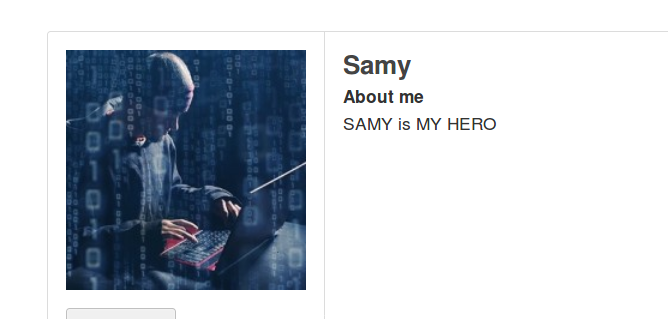


Alice’s profile after the attack:

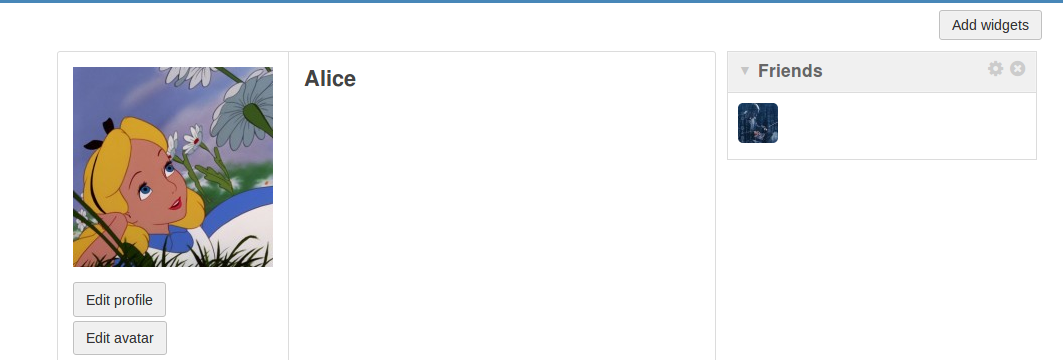


Question 3: Why do we need Line ➀? Remove this line, and repeat your attack.

If we comment this line, the attack is done on SAMY himself. When other users visit his profile, the attack won’t affect others:



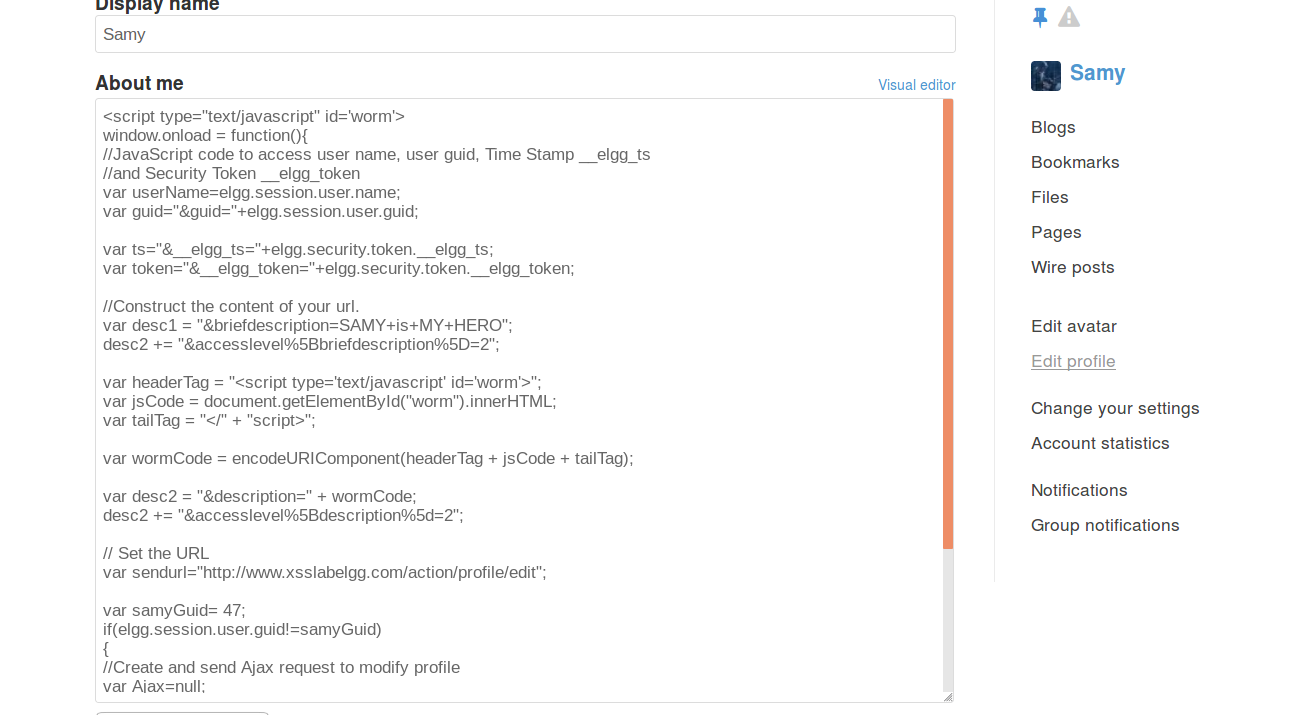
We can see that when Alice visit Samy’s profile, there is no affect on her profile:



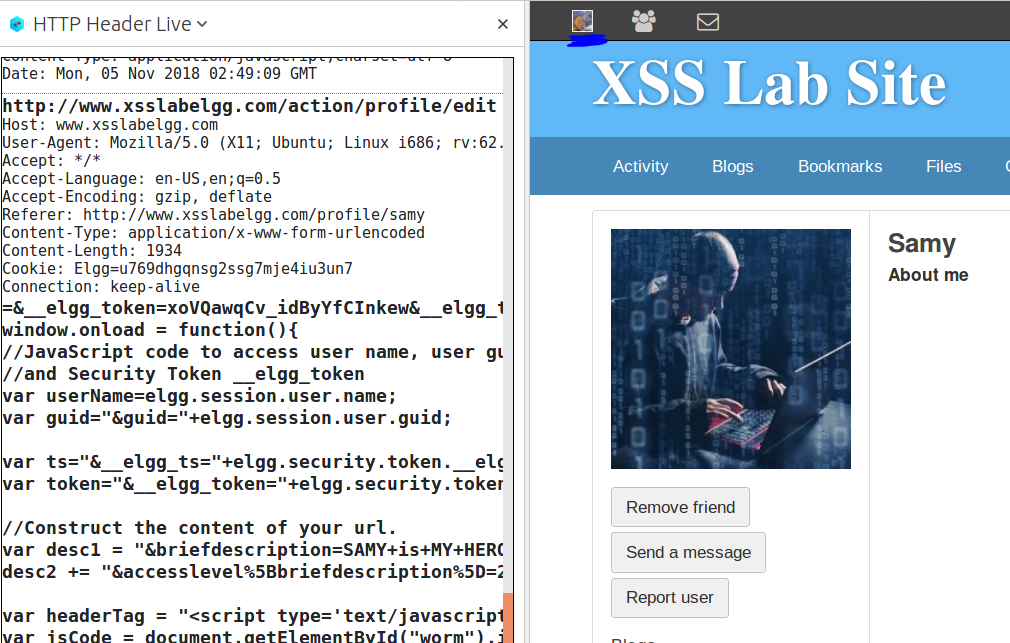
**Task 6: Writing a Self-Propagating XSS Worm**

**DOM Approach:**

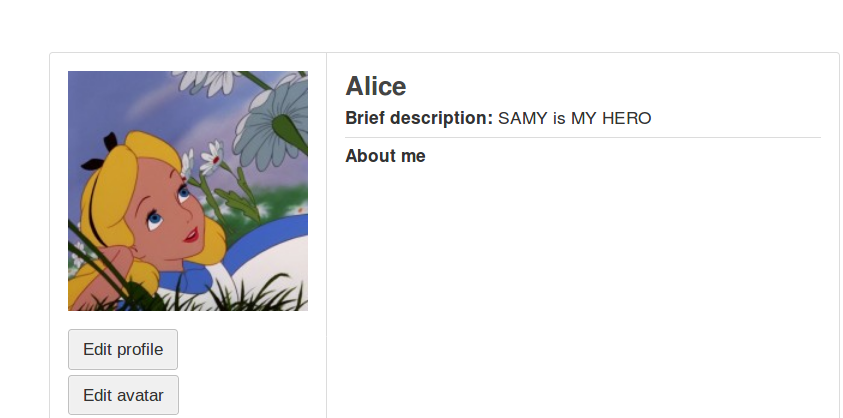
We put the malicious self-propagating XSS worm code in About me section of Samy’s profile:



When a victim visits Samy’s profile, we can see that the script code is executed and HTTP request is sent to modify brief description section to have ‘SAMY is MY HERO’ profile along with the innerHTML to include self propagating in the About Me section of the victim’s (Alice’s) profile:



After visiting Samy’s profile, we can see that in Alice’s profile the brief description is added and also the About me section is added. Here, the script code is not displayed in the web page:

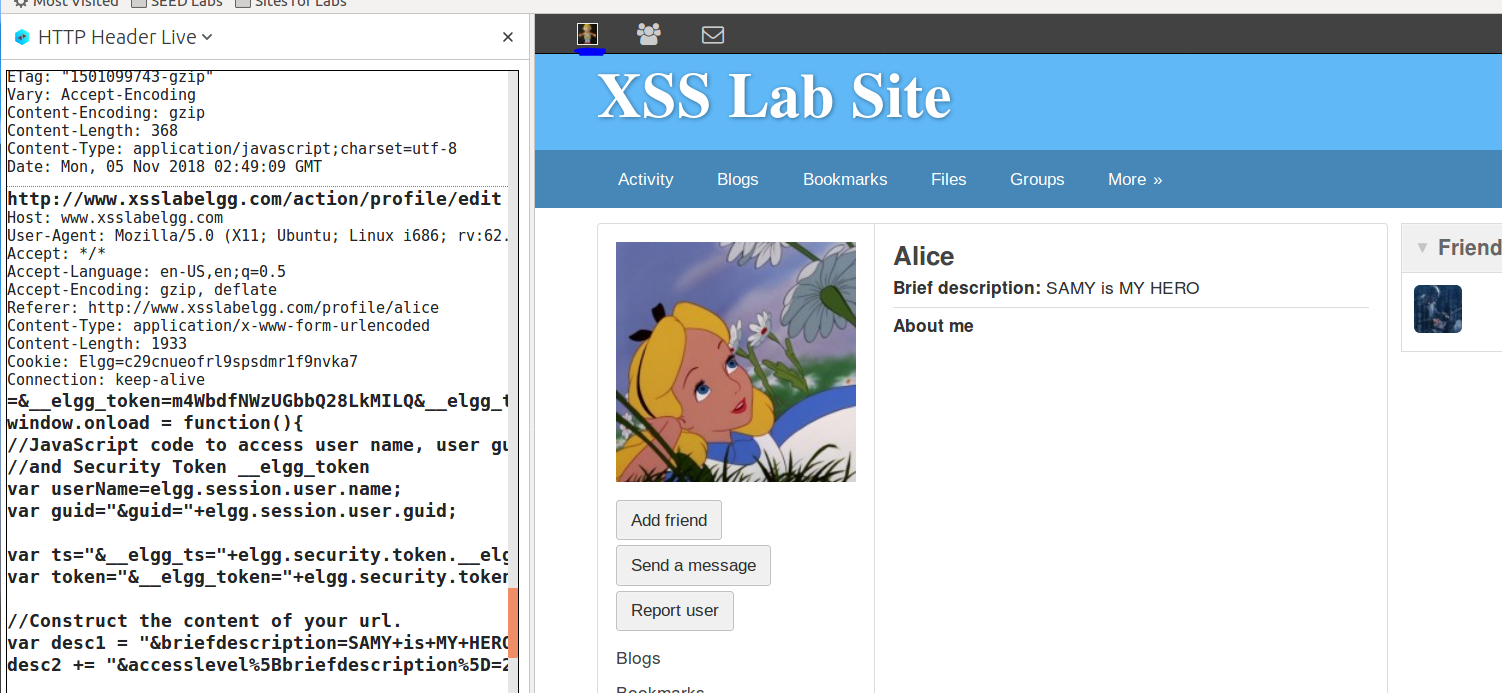


Similarly for Boby:

Boby’s profile before the attack:



When Boby visit’s Alice’s profile:

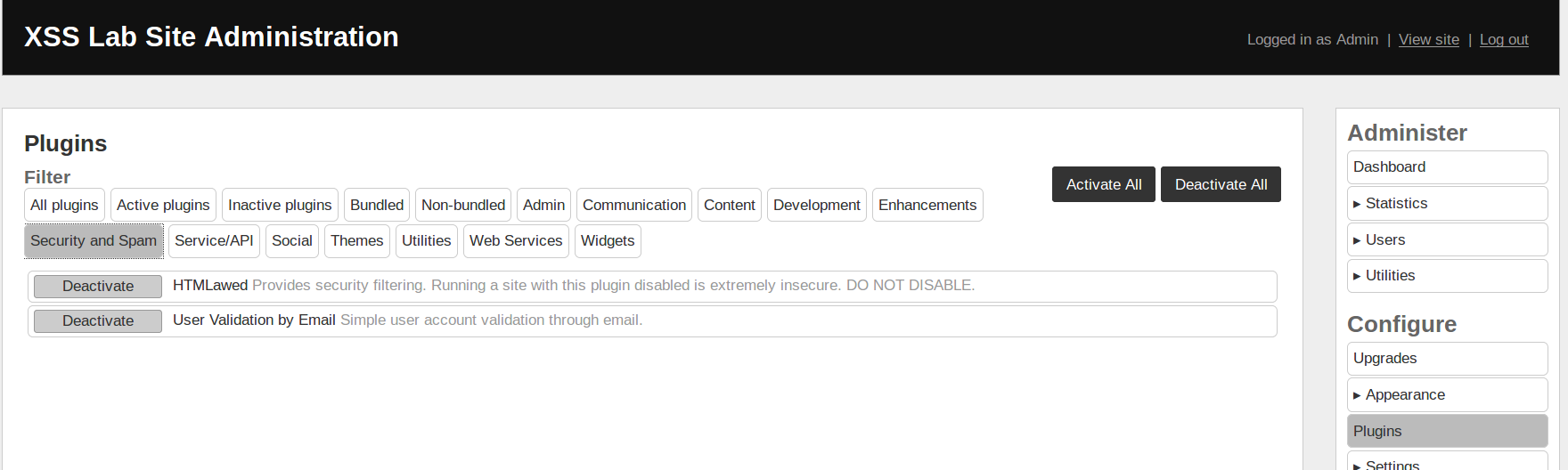


Boby’s brief description and About me fields have changed:

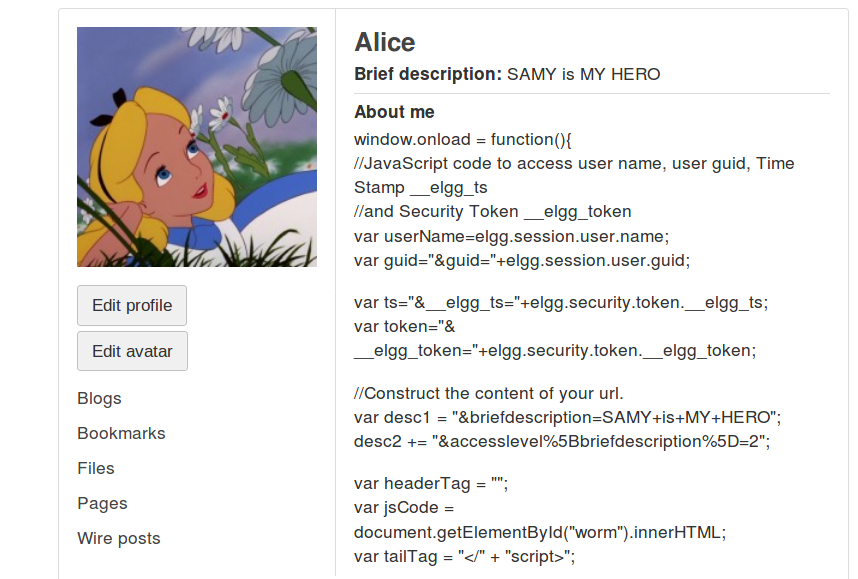


**Task 7: Countermeasures:**

Activating the HTML Lawed:



We can see that when this is activated, the malicious script tag is removed and the entire content is shown in the profile. The attack does not work as there is no script to execute.

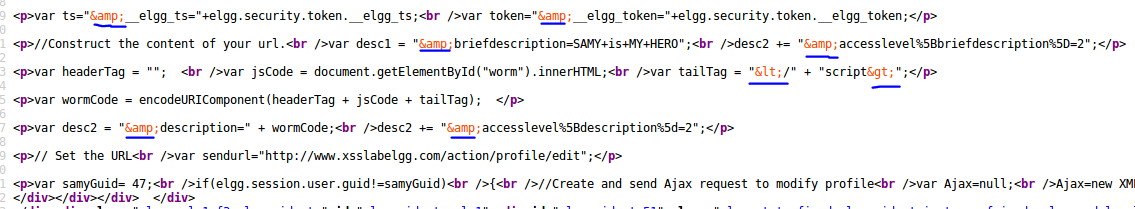


Commenting, all the specified lines in all the files, we can see that the special characters used in HTML are converted and replaced by their equivalents:

& = &amp

< = &lt

> = &gt



This way the attack won’t work as the tags cannot be used to attack.