INFORMATION SECURITY LABORATORY

WEEK 8: XSS ATTACKS

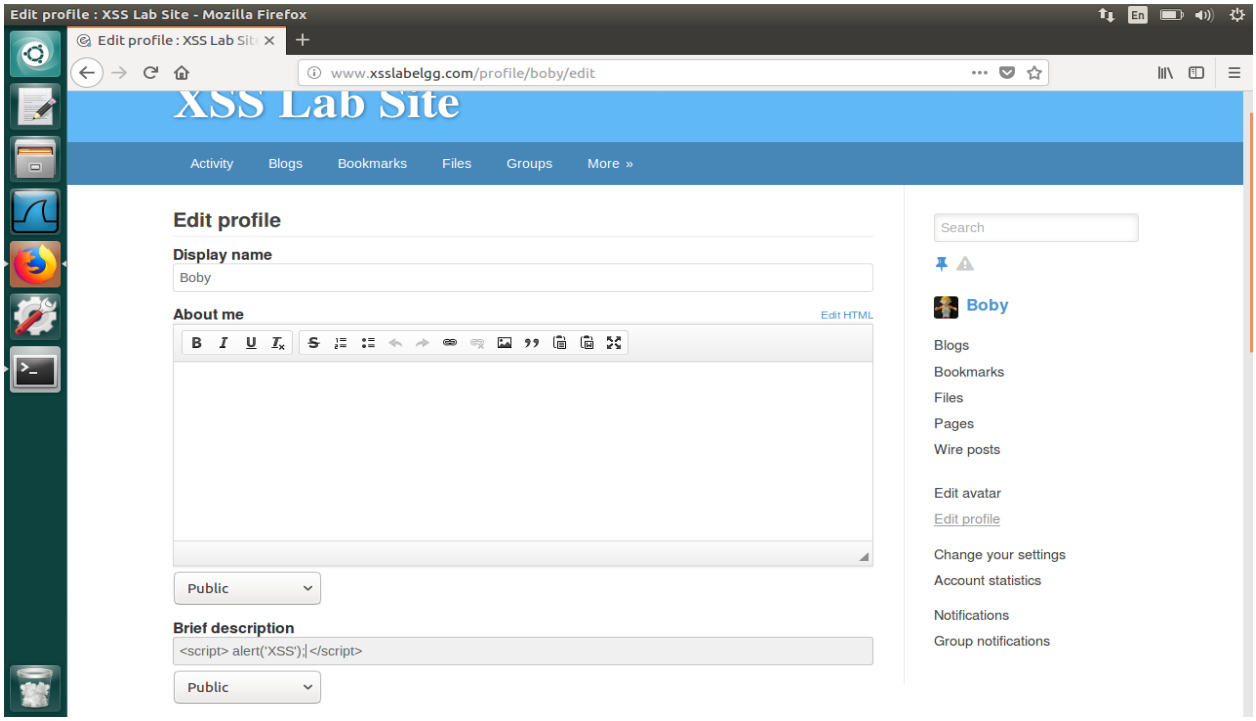
BY: RAJDEEP SENGUPTA

SRN: PES1201800144

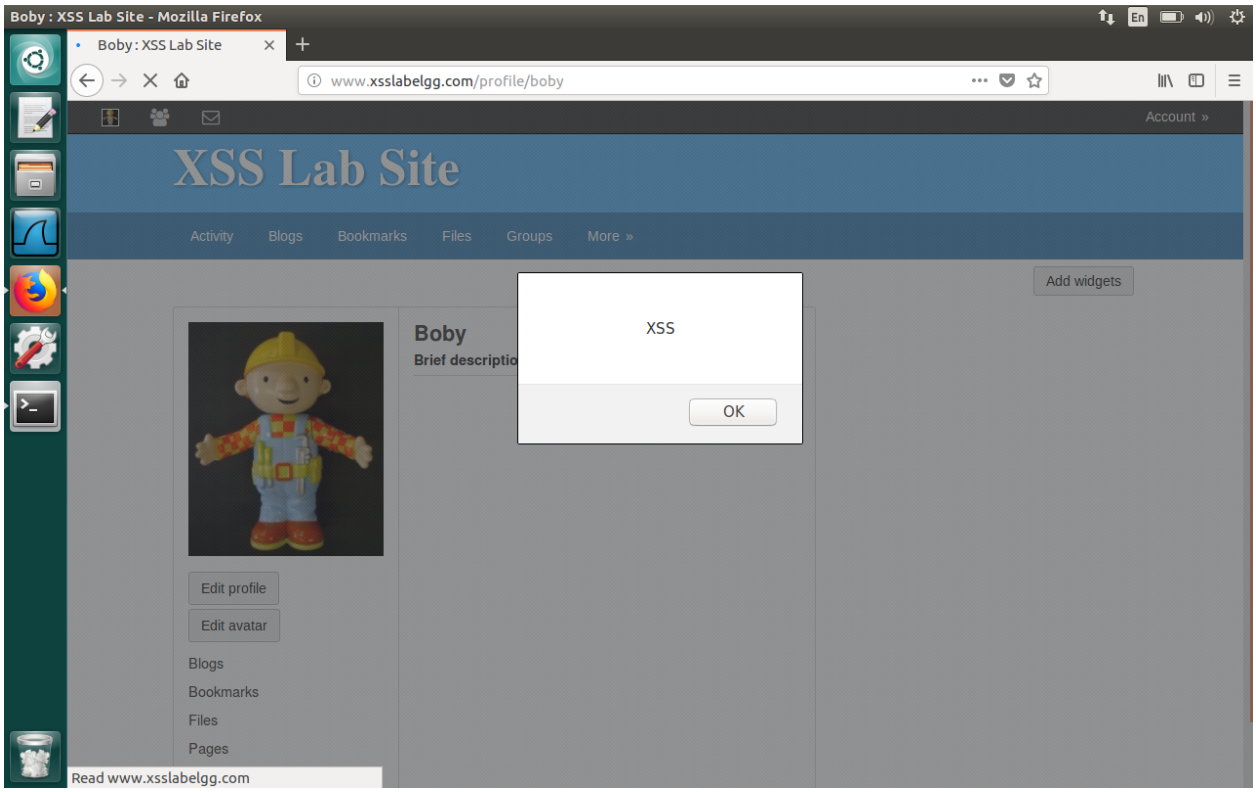
SECTION: C

Note: Please find the terminal username as my SRN followed by my name VM@pes1201800144rajdeep'. Also find the description of each task following the screenshots.

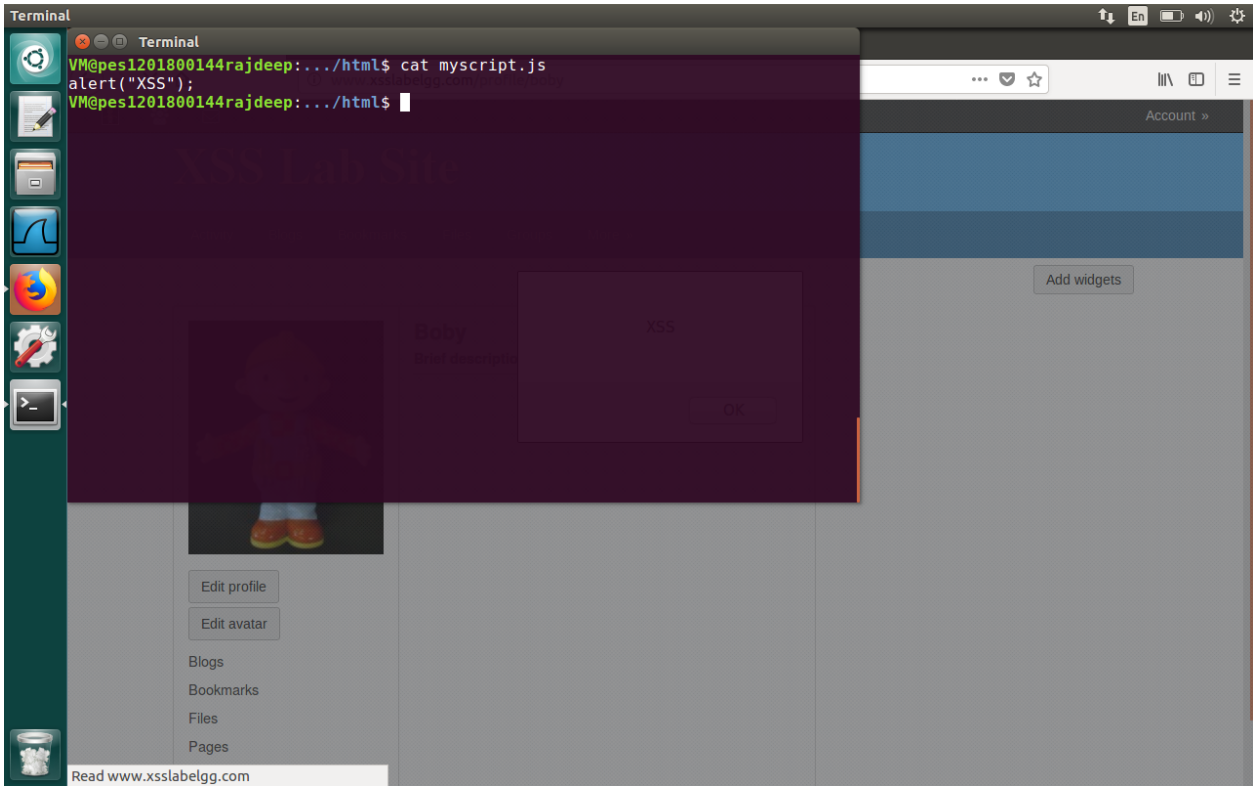
TASK 1



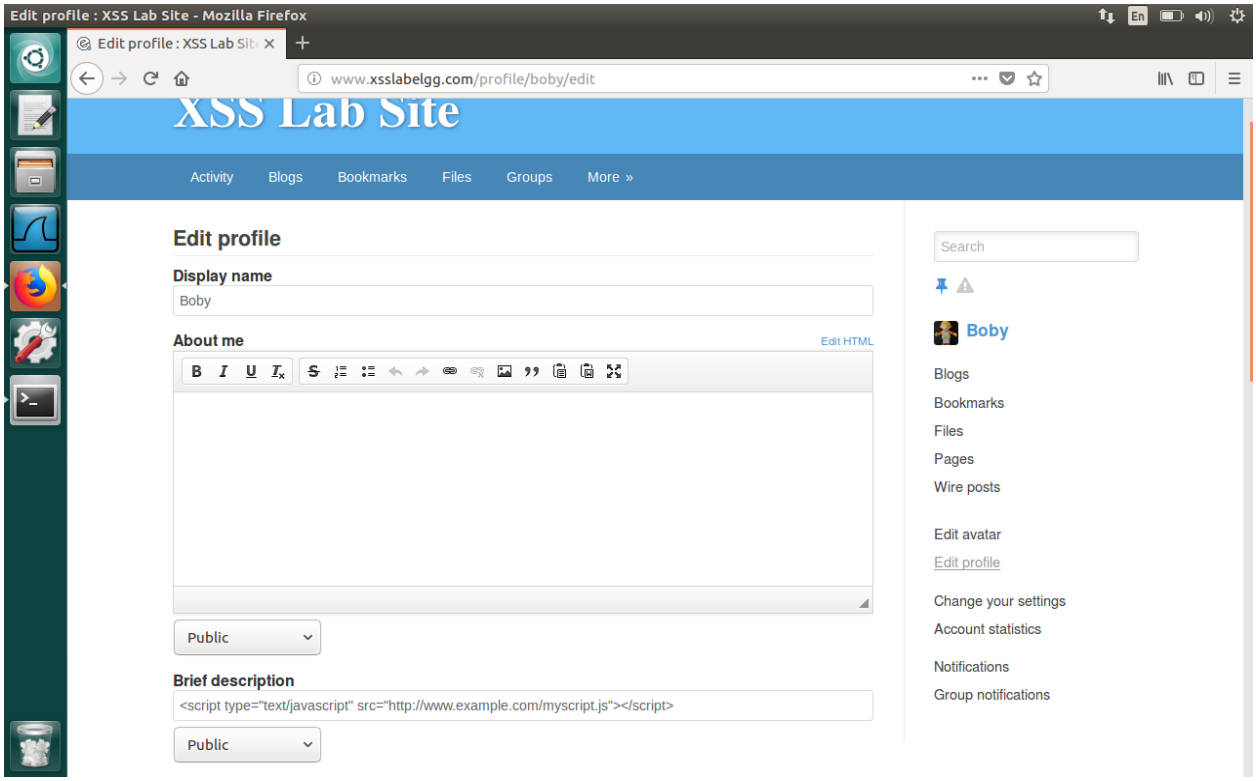
Screenshot 1.1: Adding the javascript code in the description of Boby's profile



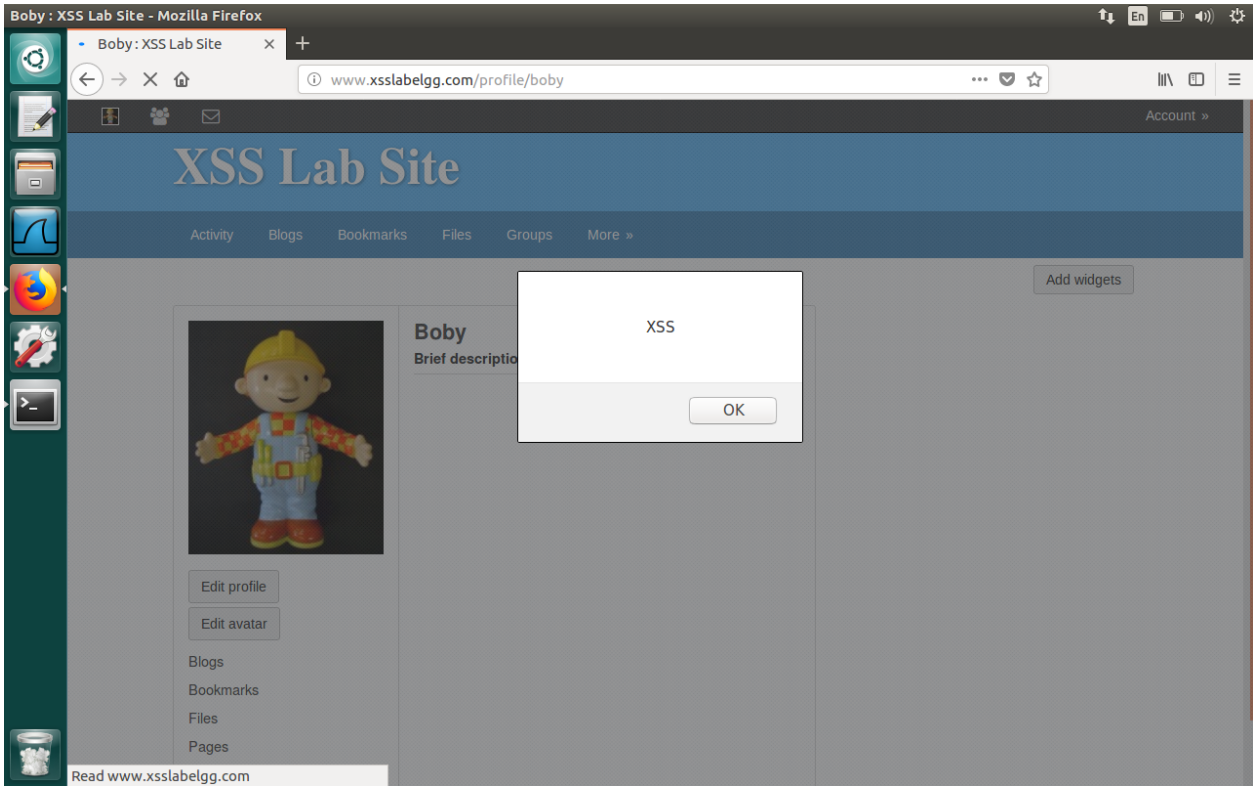
Screenshot 1.2: Whenever profile page is loaded, the javascript code works and alert message is displayed



Screenshot 1.3: Adding the javascript code in /var/www/html/myscript.js



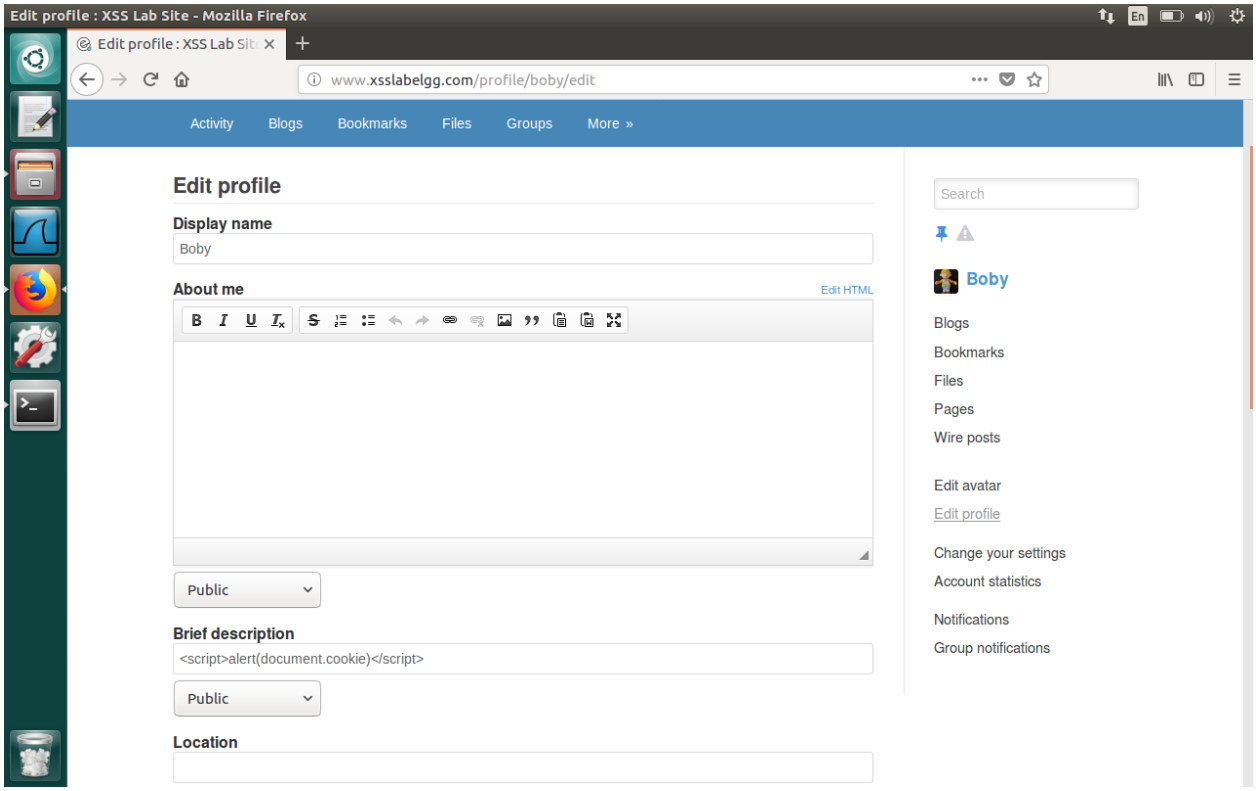
Screenshot 1.4: Adding the script source to the location of file since example.com refers to /var/www/html/ directory that has already been set in /etc/hosts file



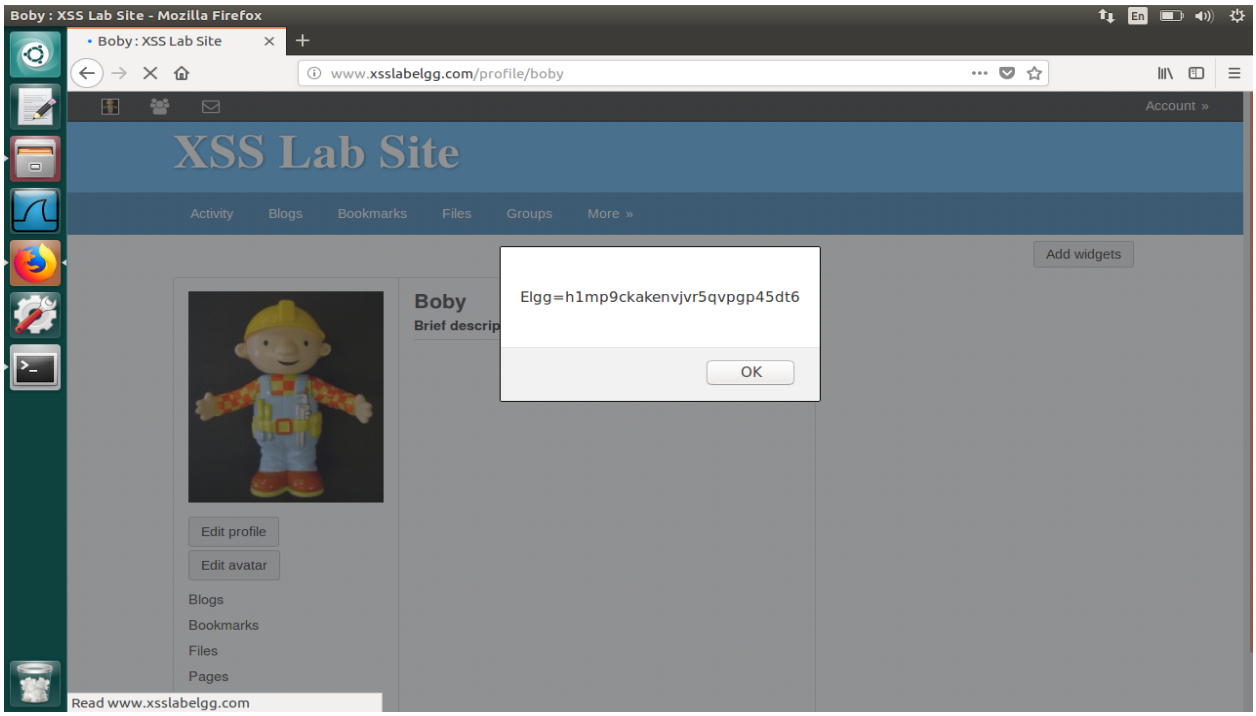
Screenshot 1.5: Javascript code works successfully when placed in /var/www/html/ directory

When the profile page loads, the javascript code given in the profile description is executed. So, in the description, any javascript code can be given which can be malicious. This can also be done by referencing to a .js file on some server(generally attacker's machine).

TASK 2:



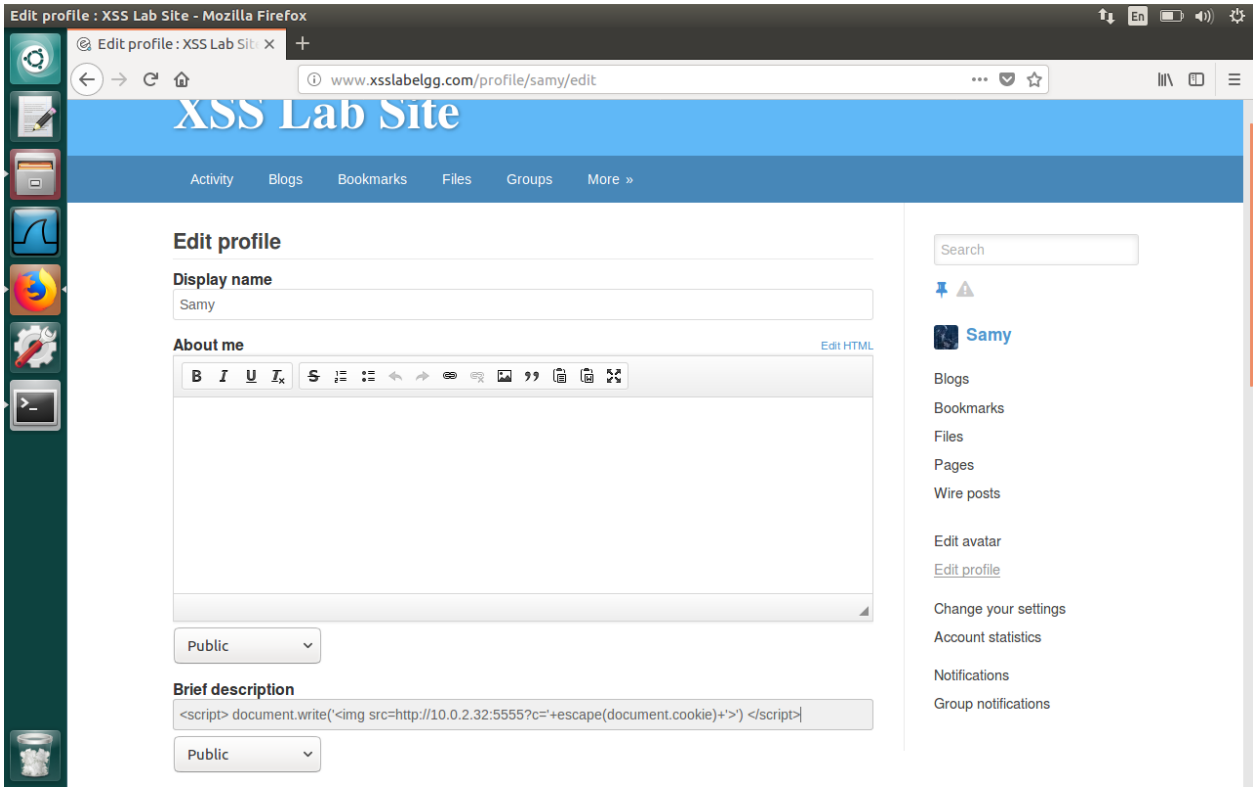
Screenshot 2.1: Fetching website cookie



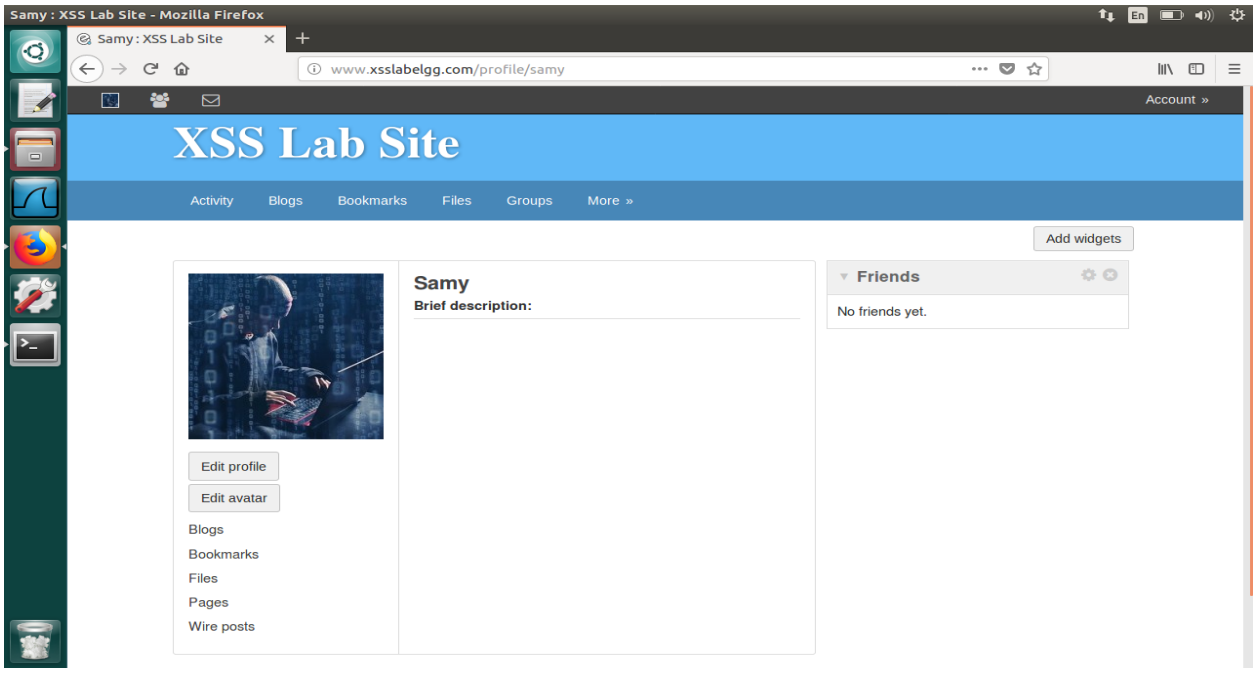
Screenshot 2.2: Successfully alerted the website cookie(Elgg token)

Private cookies of some website can easily be fetched using javascript

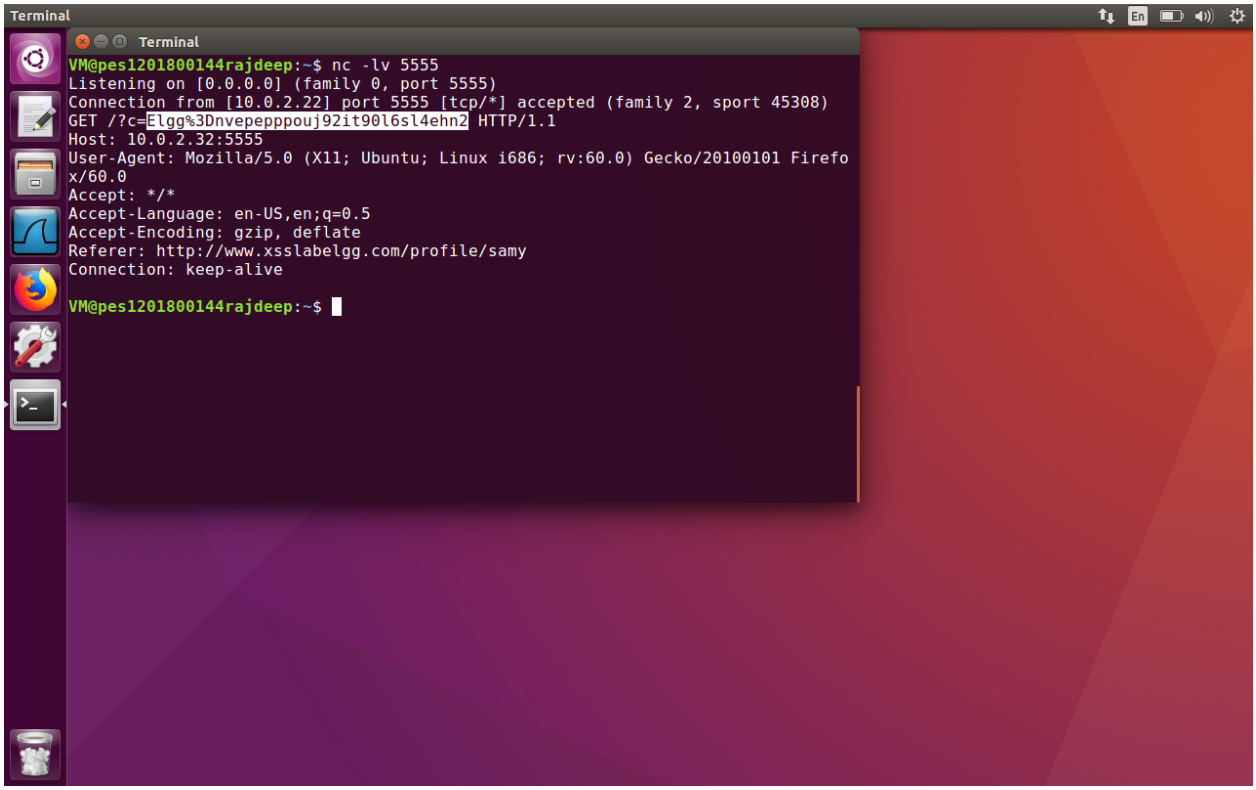
TASK 3:



Screenshot 3.1: sending webpage cookie to attacker machine with IP 10.0.2.32 on port 5555



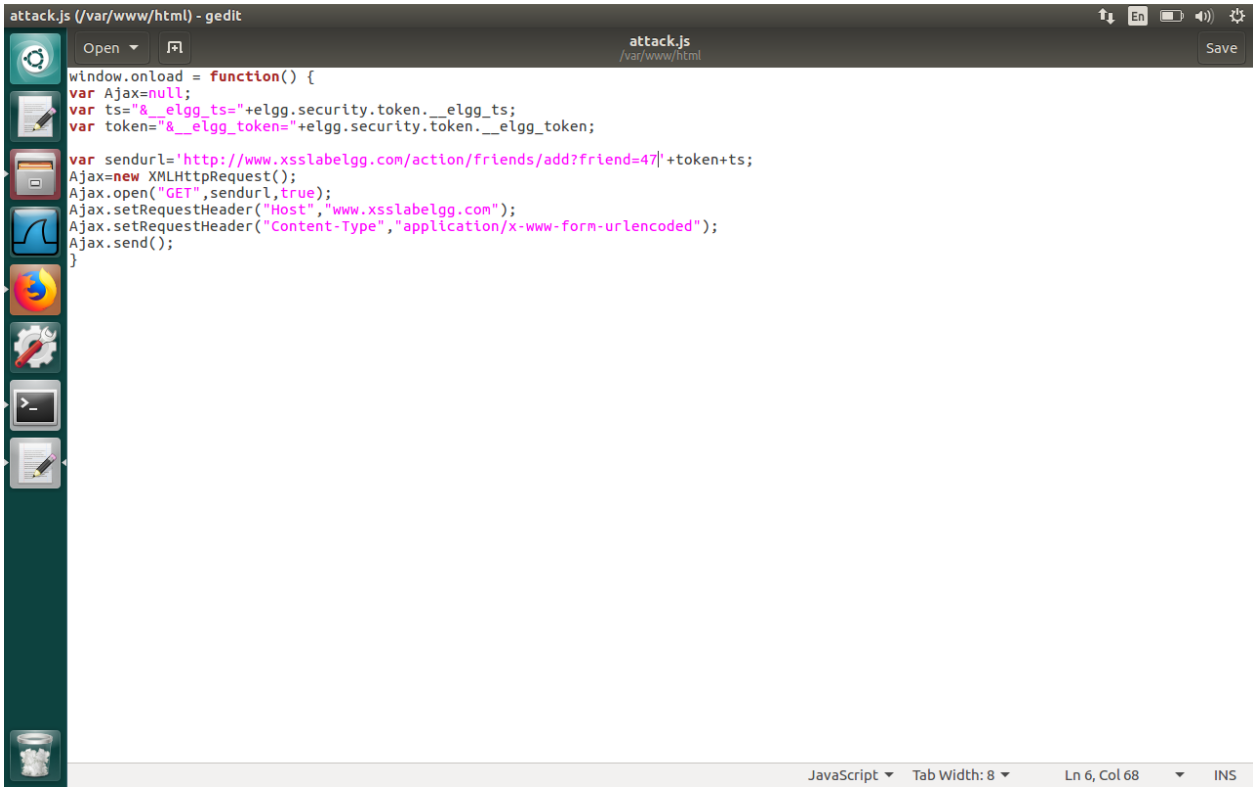
Screenshot 3.2: Profile loaded



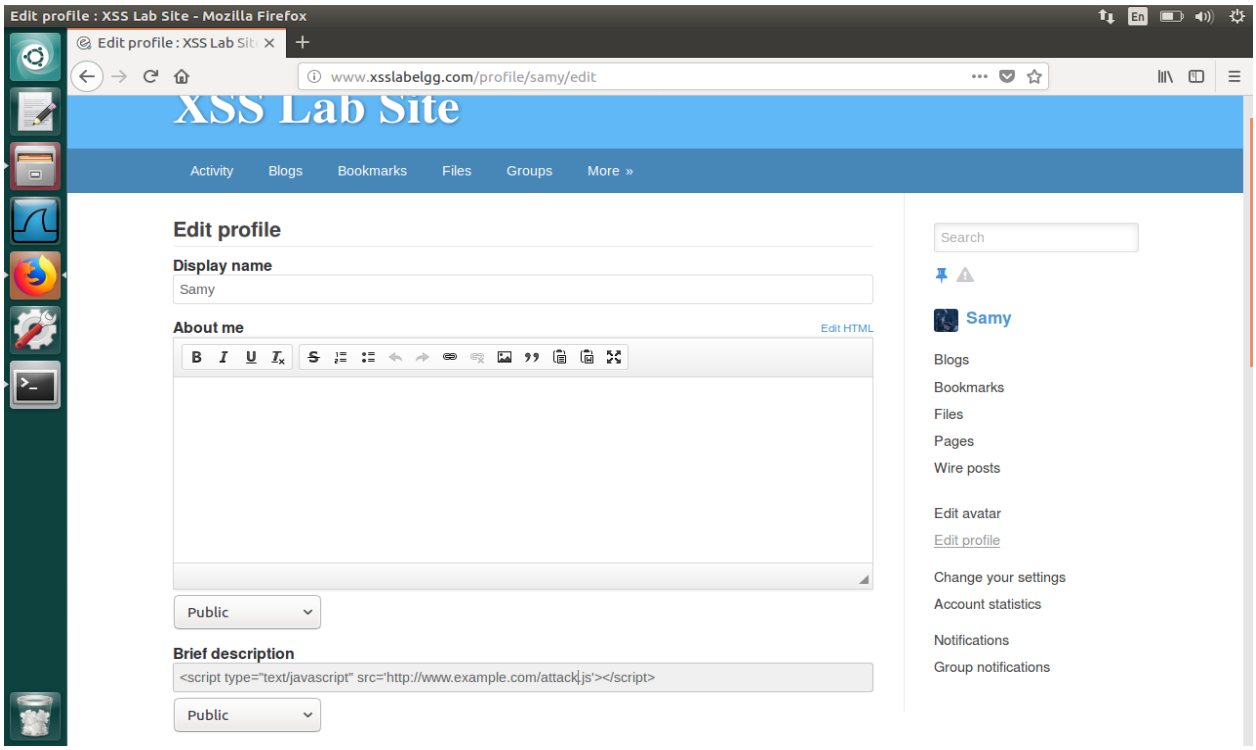
Screenshot 3.3: On attacker machine, the cookie details can be fetched while listening on port 5555 via netcat

As in the earlier task, the session cookie can easily be fetched using javascript code. In this task, the fetched cookie is sent to the attacker machine listening on netcat port 5555 as specified. In the attacker machine, it can be seen that the Elgg token is successfully captured on port 5555.

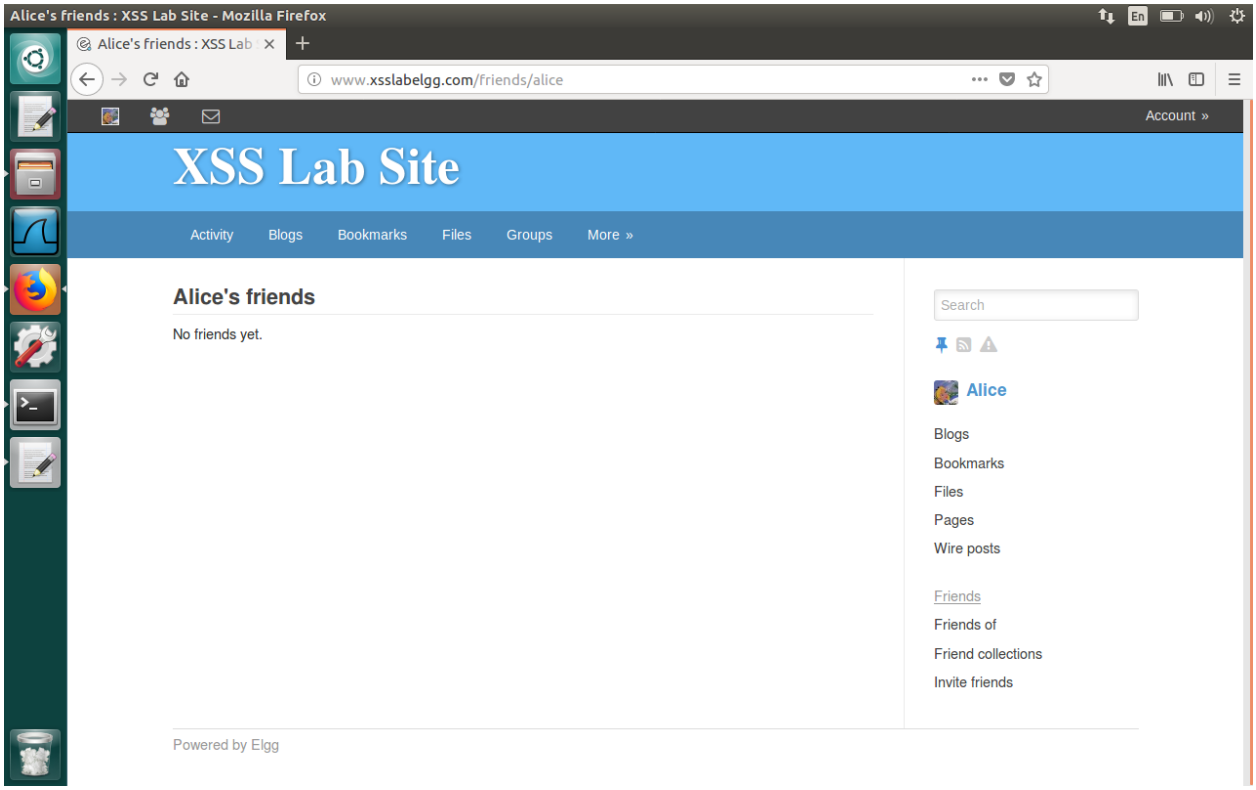
TASK 4:



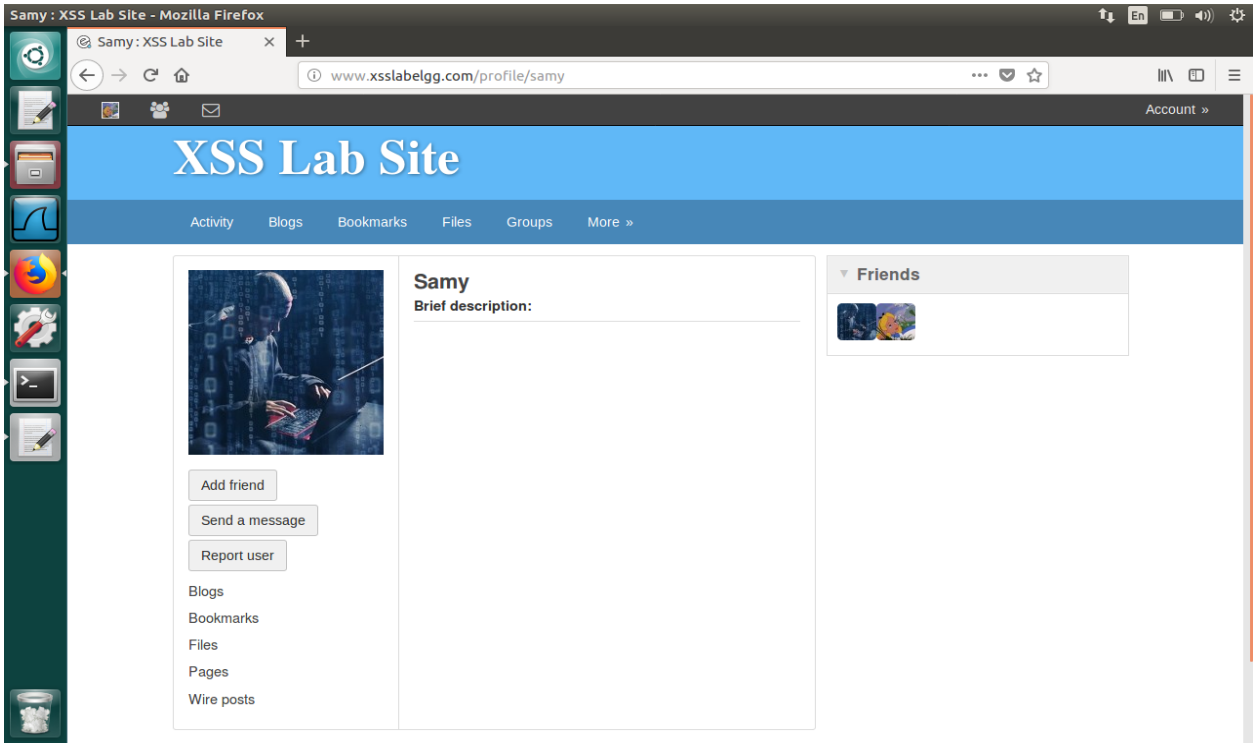
Screenshot 4.1: Attack.js file in /var/www/html



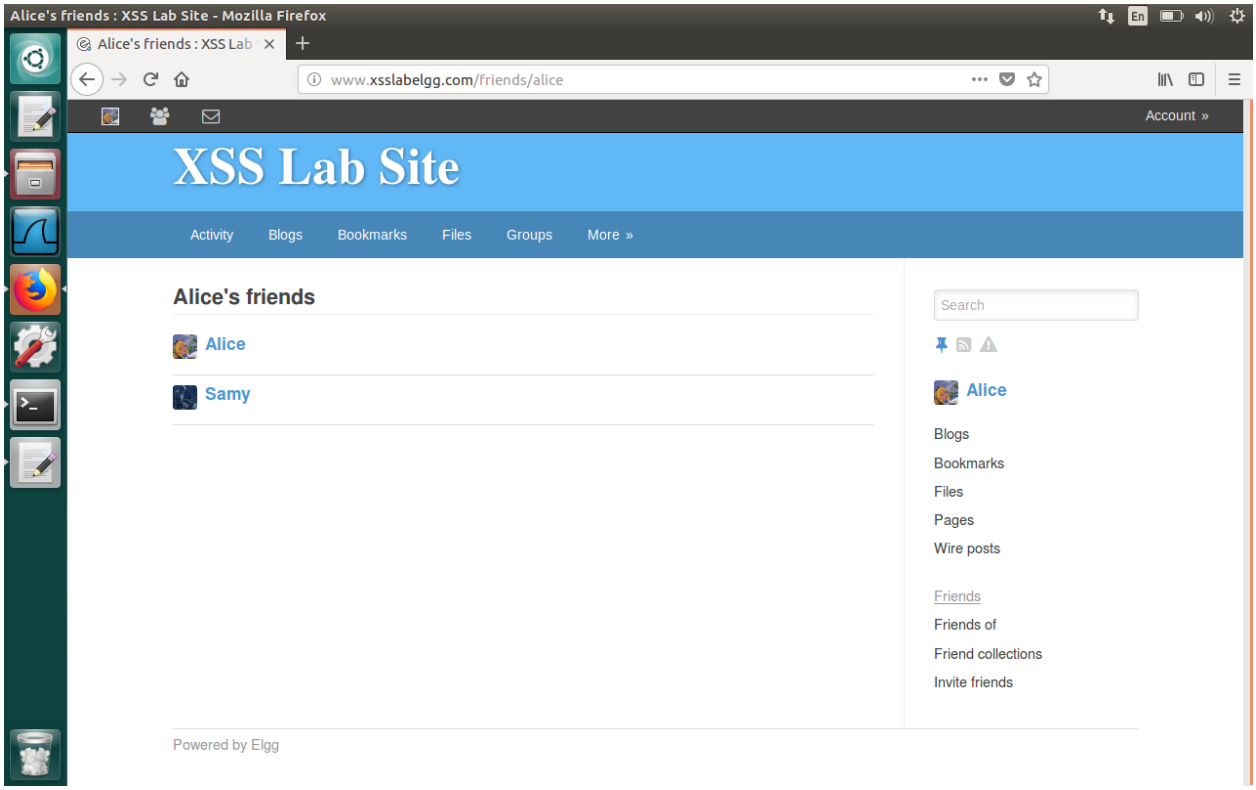
Screenshot 4.2: Adding javascript code in profile description of Samy



Screenshot 4.3: Alice logs into her profile and sees her friends before the attack



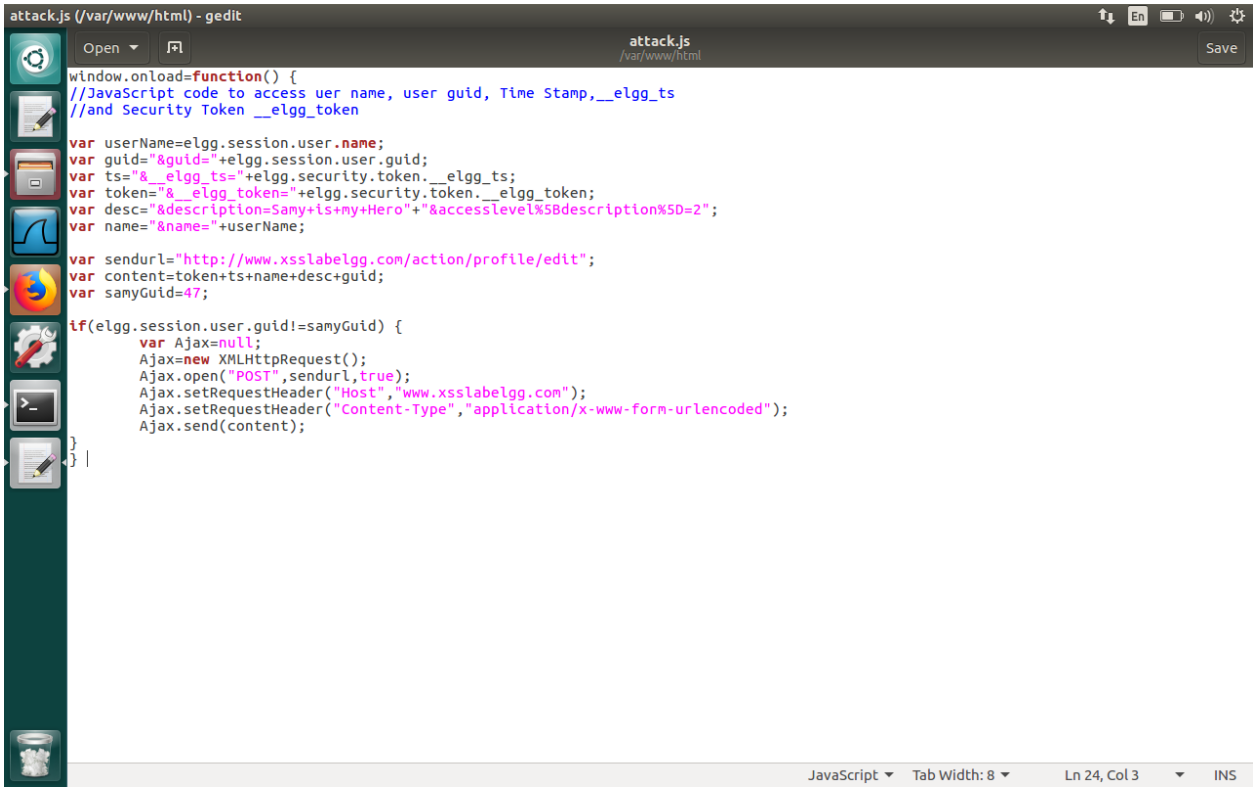
Screenshot 4.4: Alice visits Samy's profile and the javascript code is triggered in the background



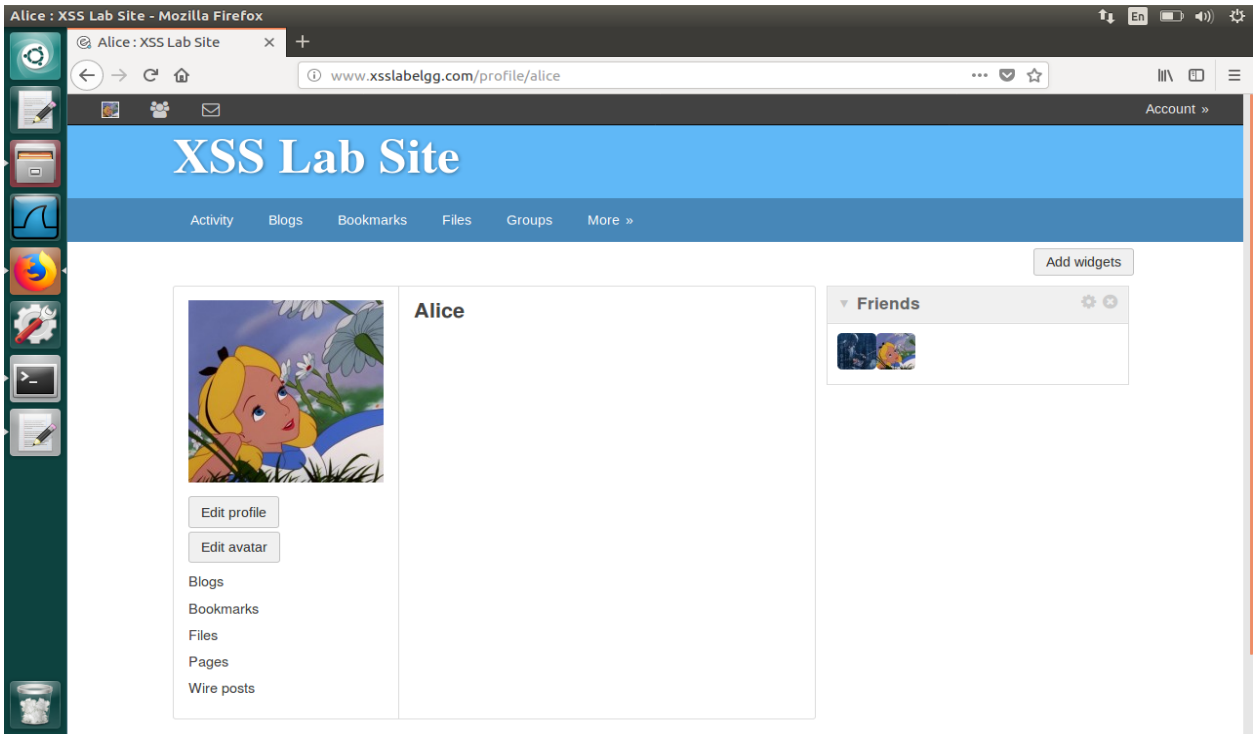
Screenshot 4.5: Samy gets added to Alice's profile after the attack which means attack is successful

The malicious code is put in attack.js file. This file fetches the Elgg token and Elgg secret token using javascript and sends the GET request to add friend. When Alice or any other person visits Samy's profile, the user's elgg secret token is fetched dynamically and GET request is sent to Samy to 'Add Friend'. This makes the current user a friend of Samy without their knowledge.

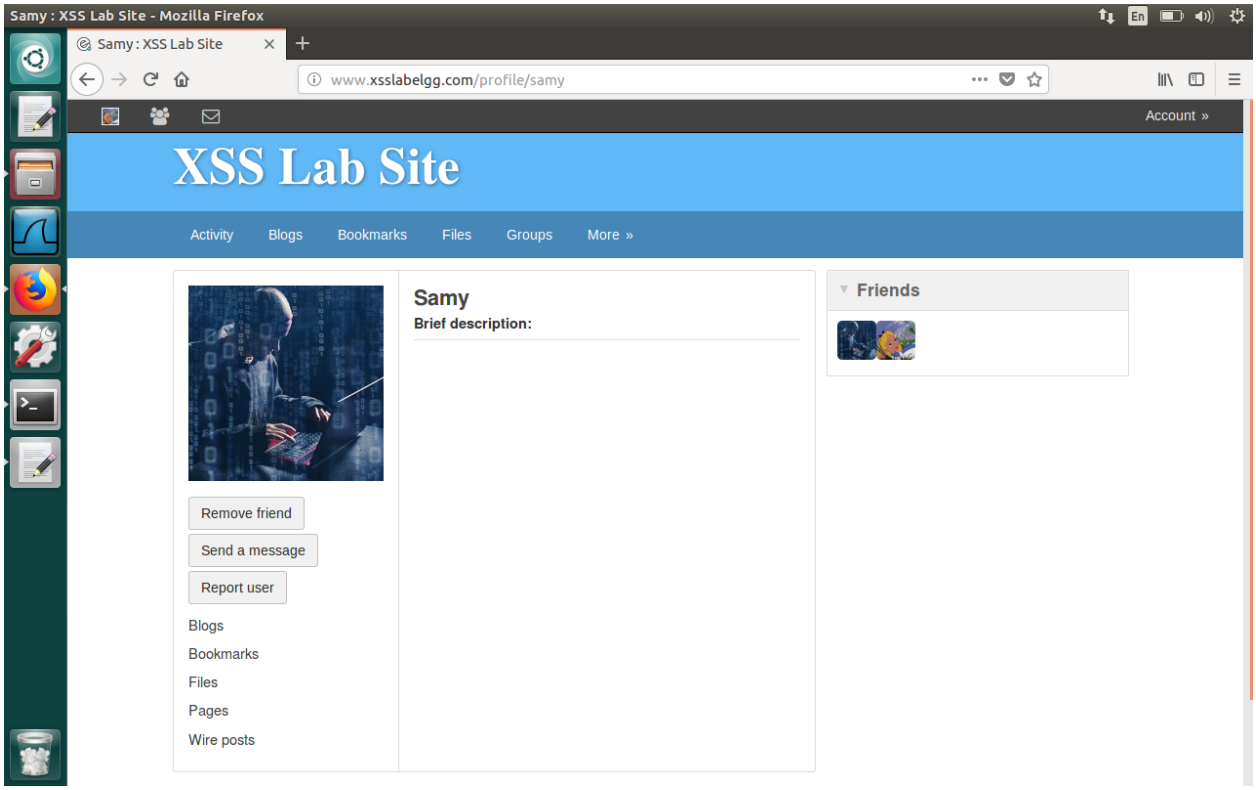
TASK 5:



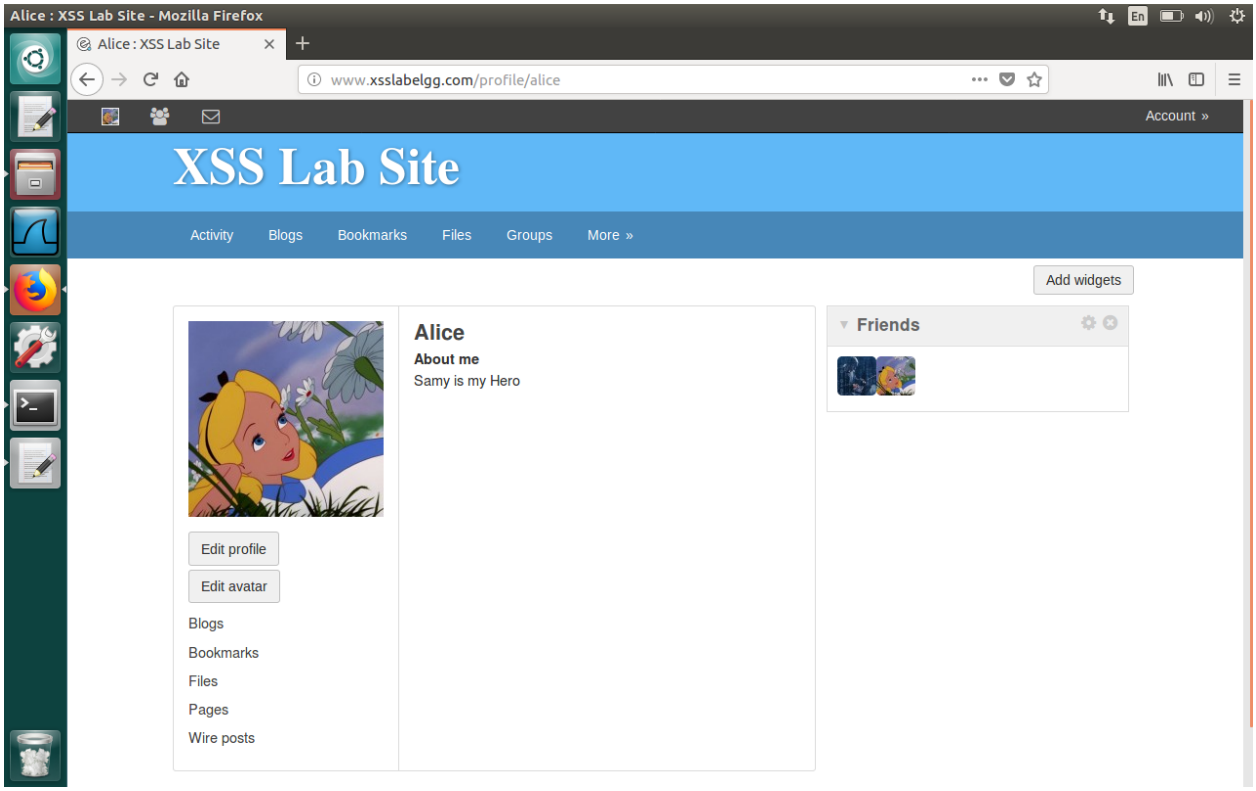
Screenshot 5.1: Attack.js file



Screenshot 5.2: Alice's profile before the attack



Screenshot 5.3: Alice visits Samy's profile

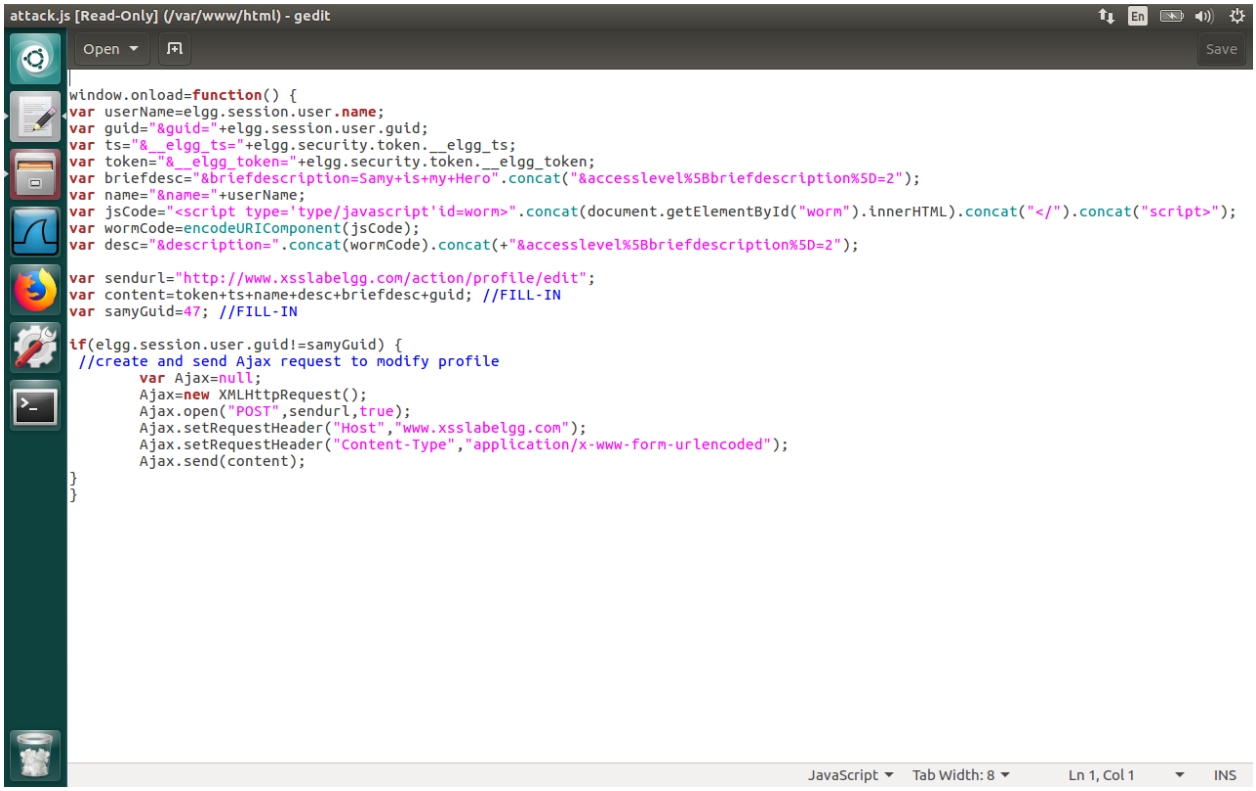


Screenshot 5.4: Alice's profile gets updated which means attack successful

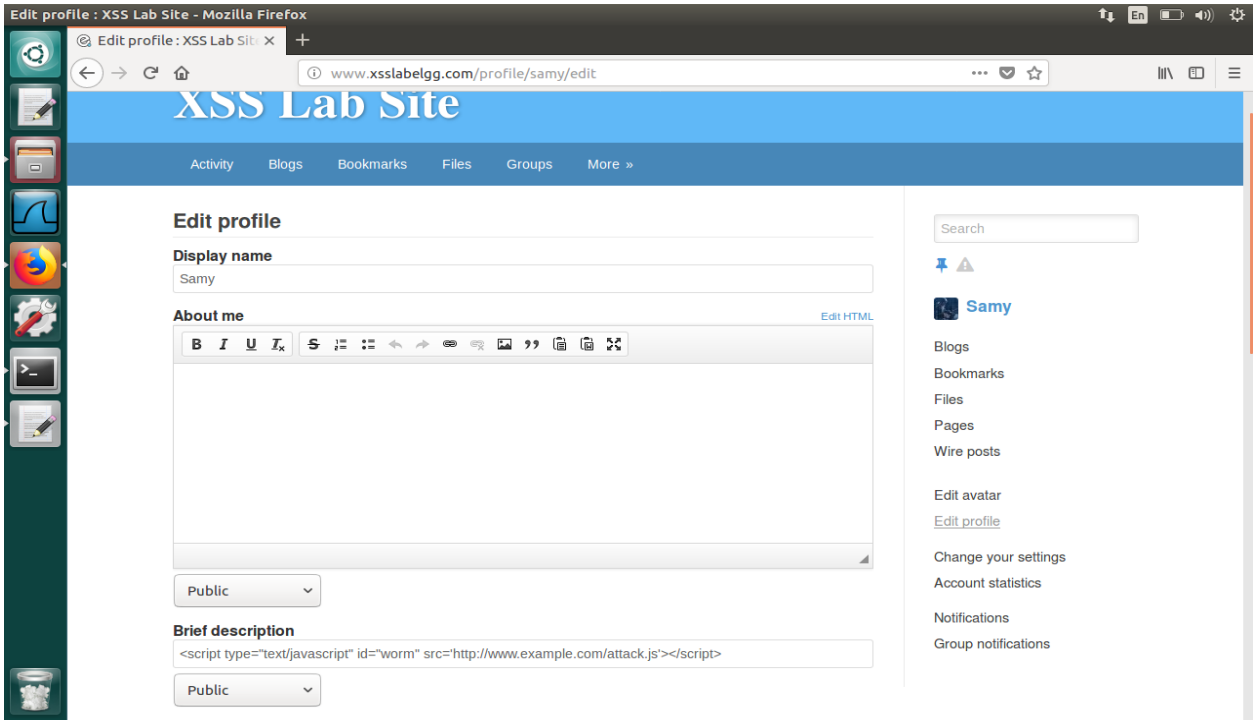
In this attack, the same is done except this time, a POST request is sent using the attack.js file.

The malicious code is put in attack.js file. This file fetches the Elgg token and Elgg secret token using javascript and sends the POST request to edit the profile description. When Alice or any other person visits Samy's profile, the user's elgg secret token is fetched dynamically and POST request is sent to change the current user's profile description to whatever message is specified in the attack.js file. In this attack, the message 'Samy is my Hero' is used as it can be seen in the code (Screenshot 5.1).

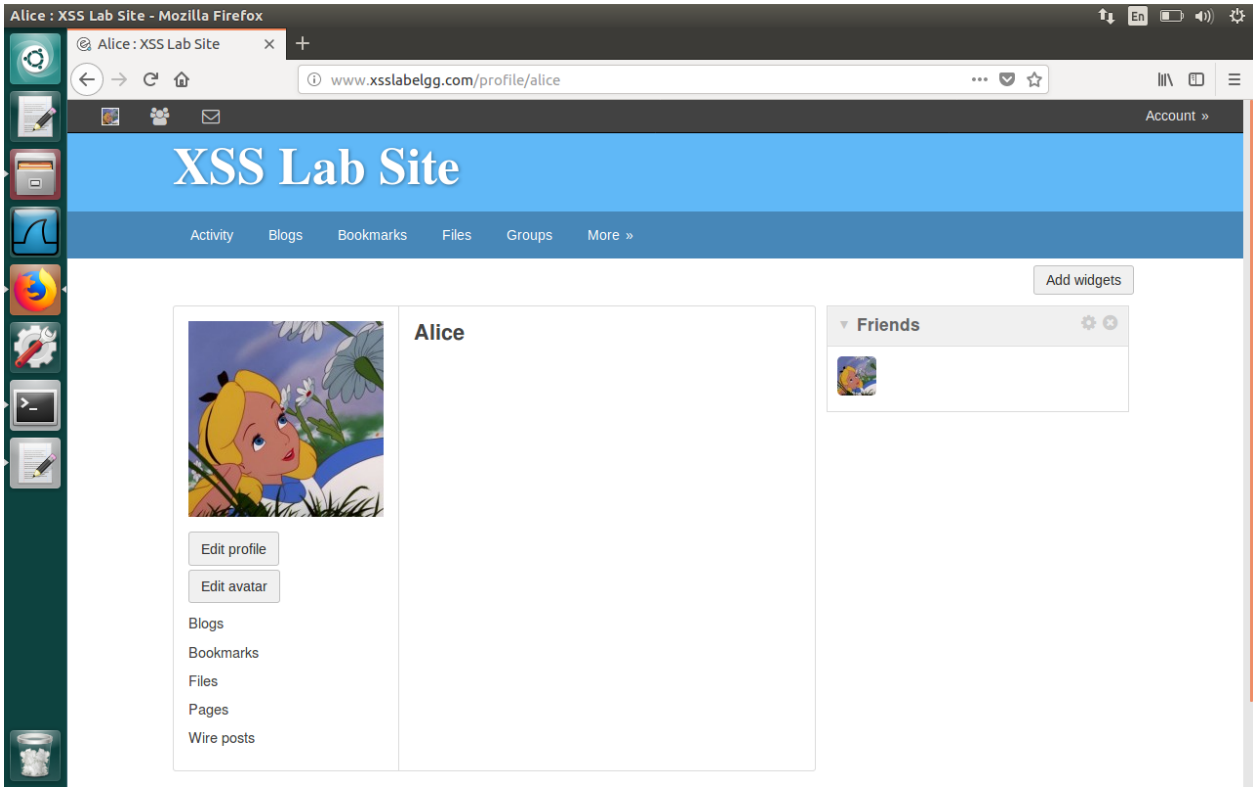
TASK 6:



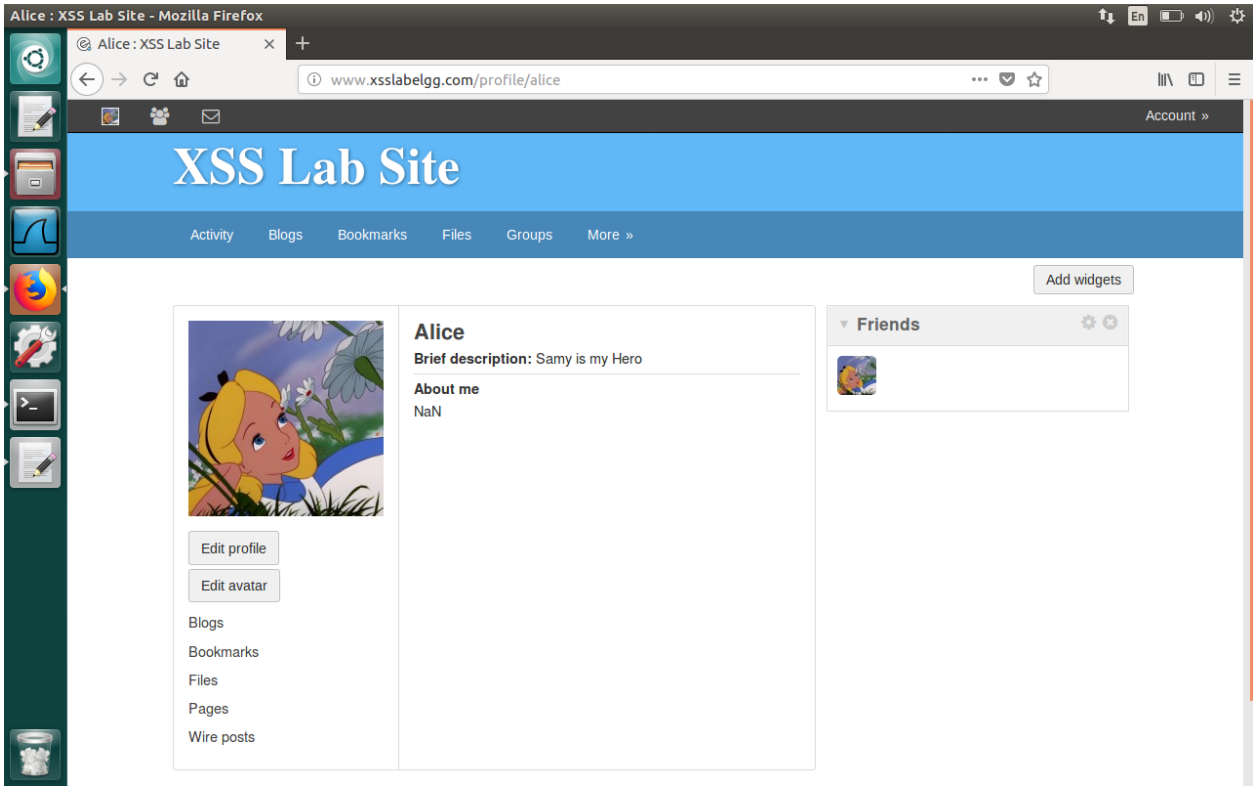
Screenshot 6.1: Attack.js code



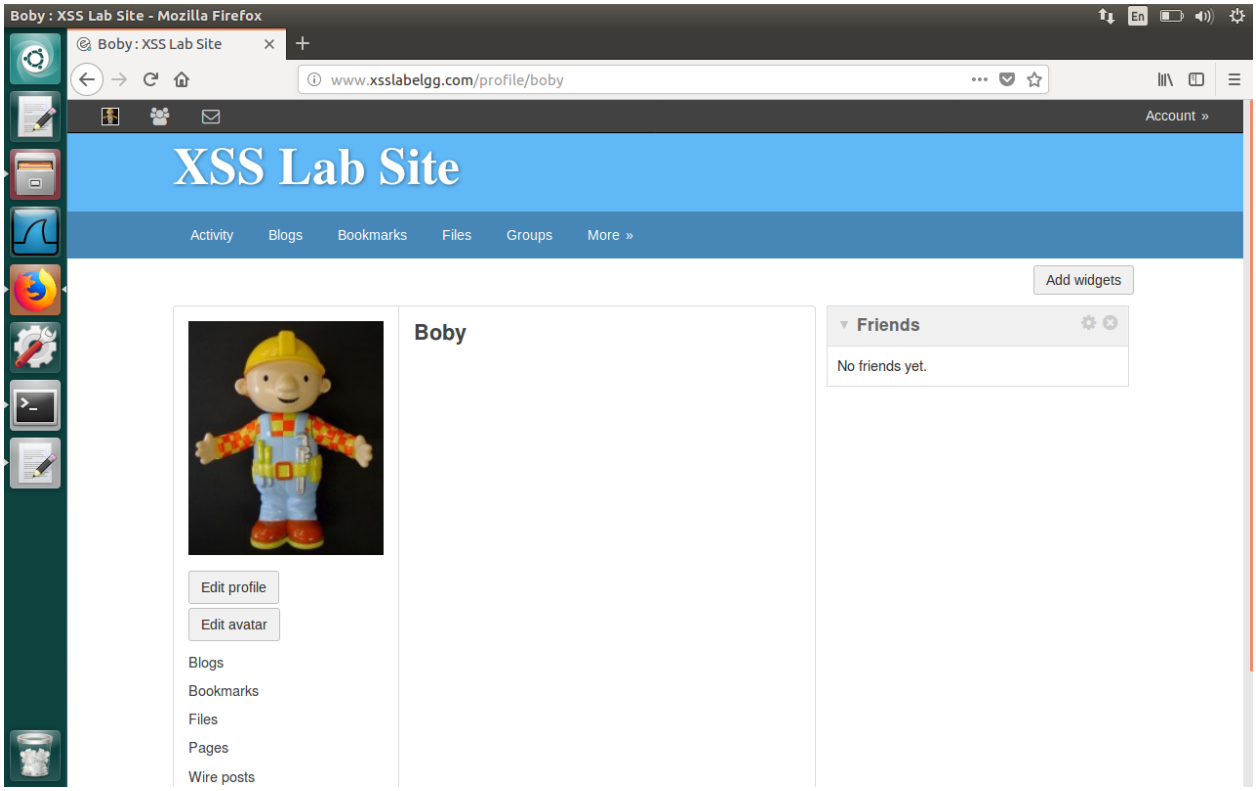
Screenshot 6.2: Adding the attack.js javascript code in description of Samy



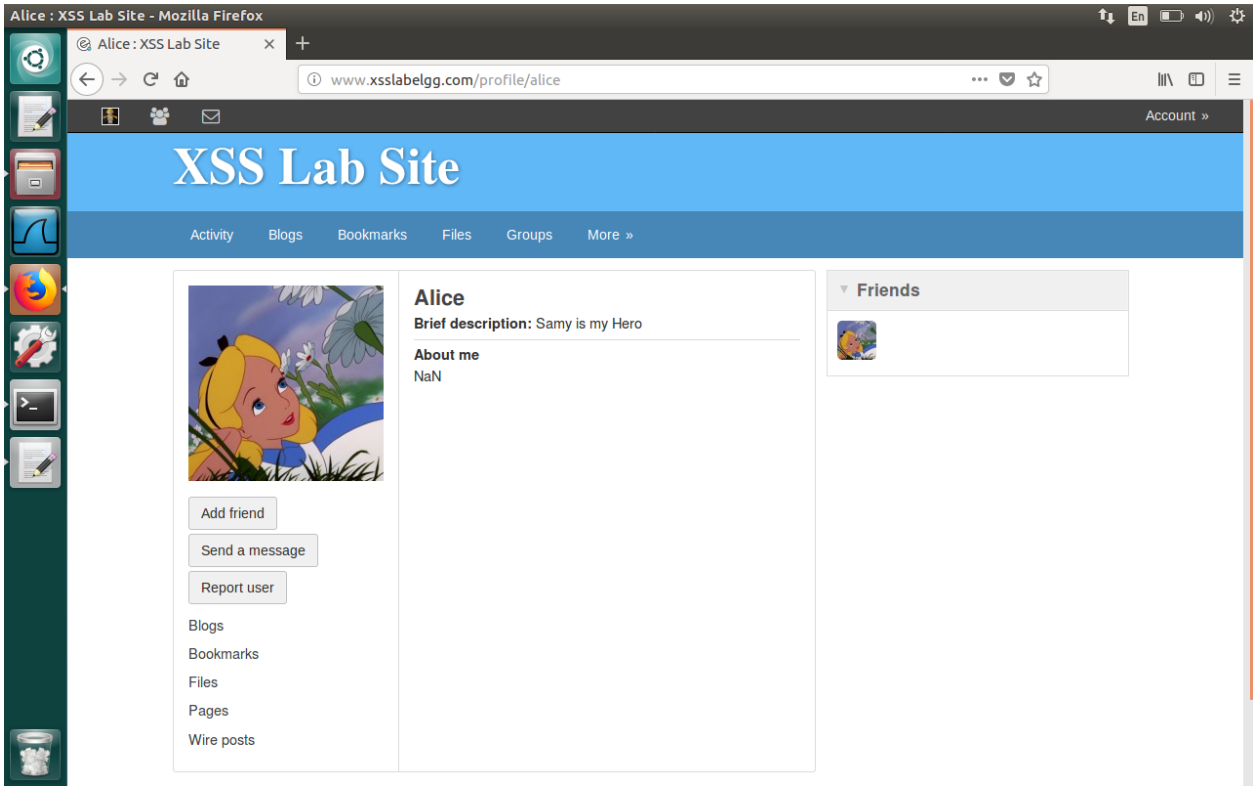
Screenshot 6.3: Alice's profile before



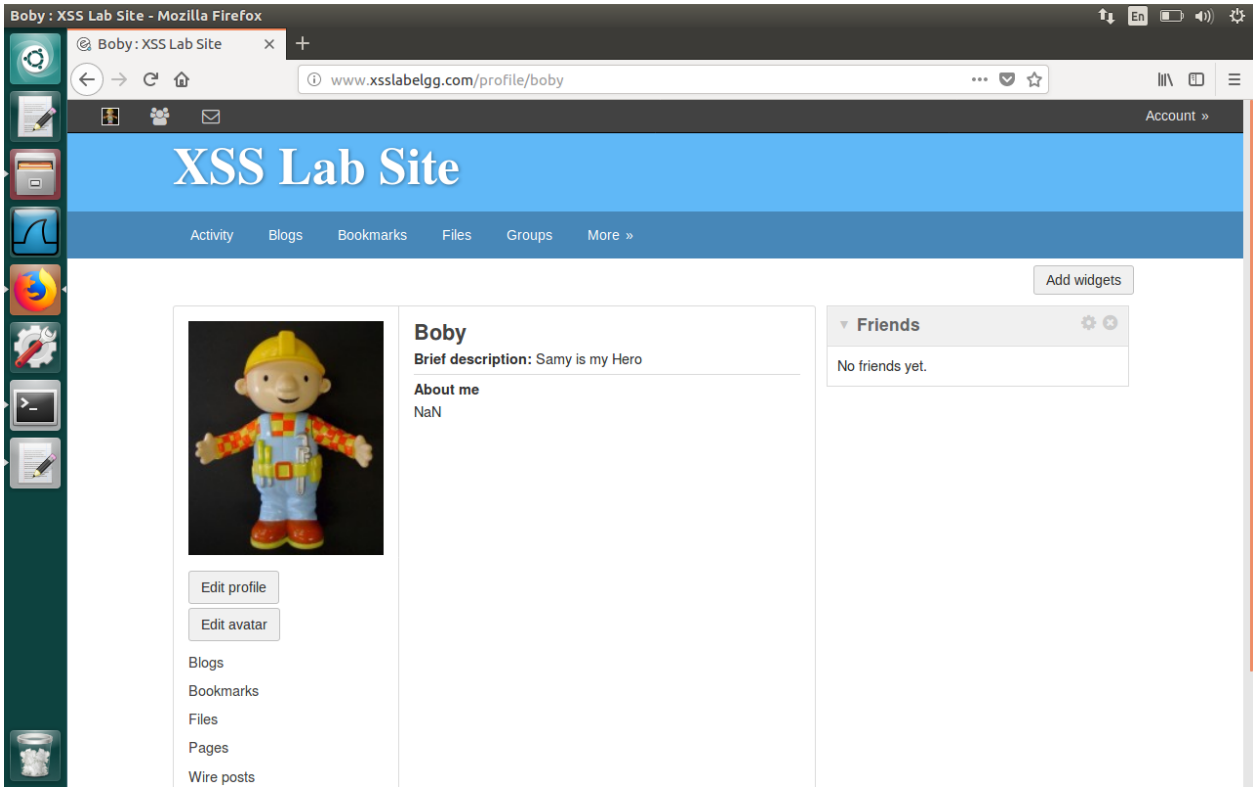
Screenshot 6.4: Alice's profile description after she visits Samy's profile



Screenshot 6.5: Boby's profile before the attack



Screenshot 6.6: Boby visits Alice's profile



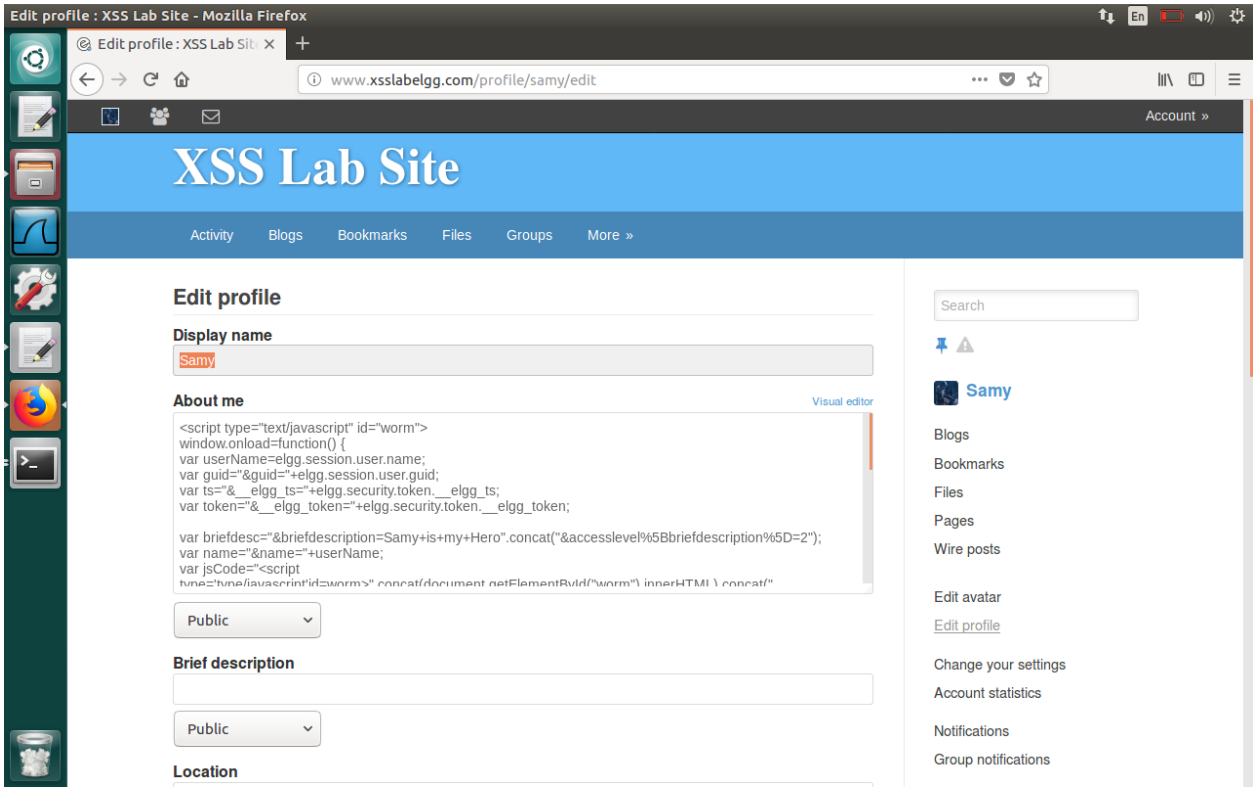
Screenshot 6.7: Boby's profile description after visiting Alice's profile

Initially Samy makes this worm only for the users visiting his profile. Alice visits Samy's profile and gets infected with the worm. Further, when Boby visits Alice(the infected victim), the worm shows its effects and infects Boby also. Hence this worm is spreading from user to user and infecting users at an exponential rate.

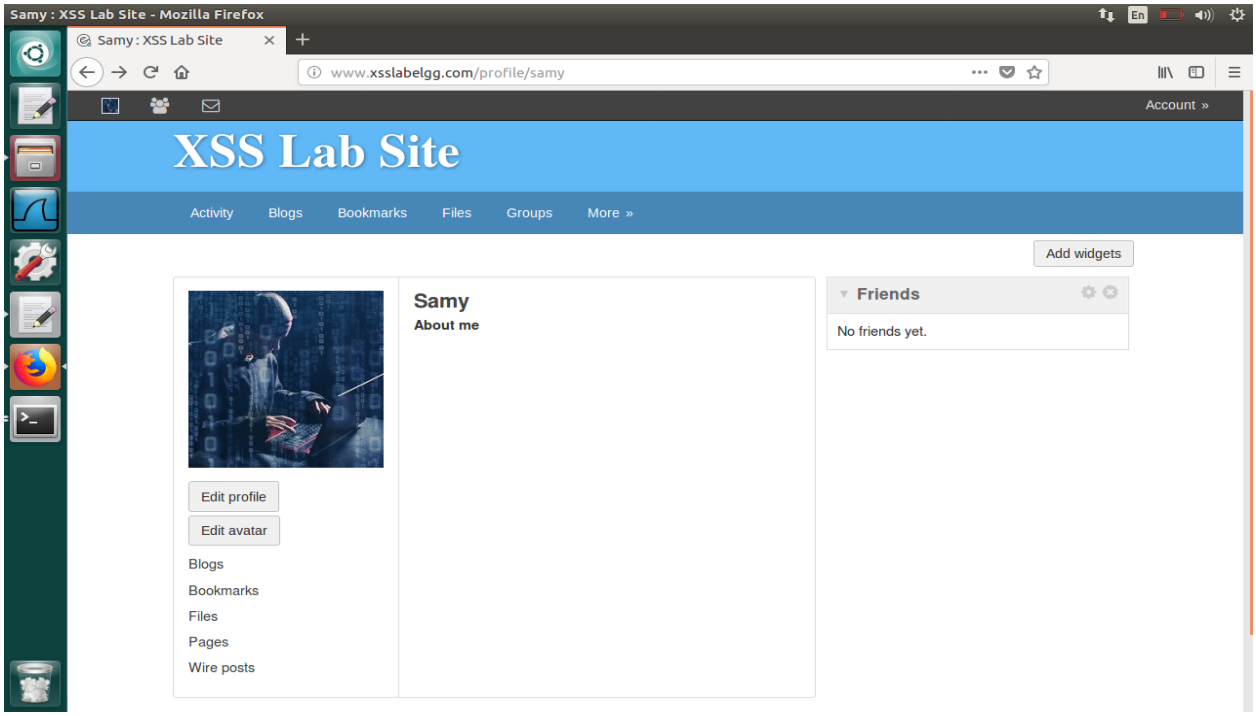
This worm is spreading by:

When a user visits the profile of the attacker, the attacker's malicious code writes itself into the description of the victim. This makes the victim like an attacker for other users. This is the mechanism it adopts to spread through.

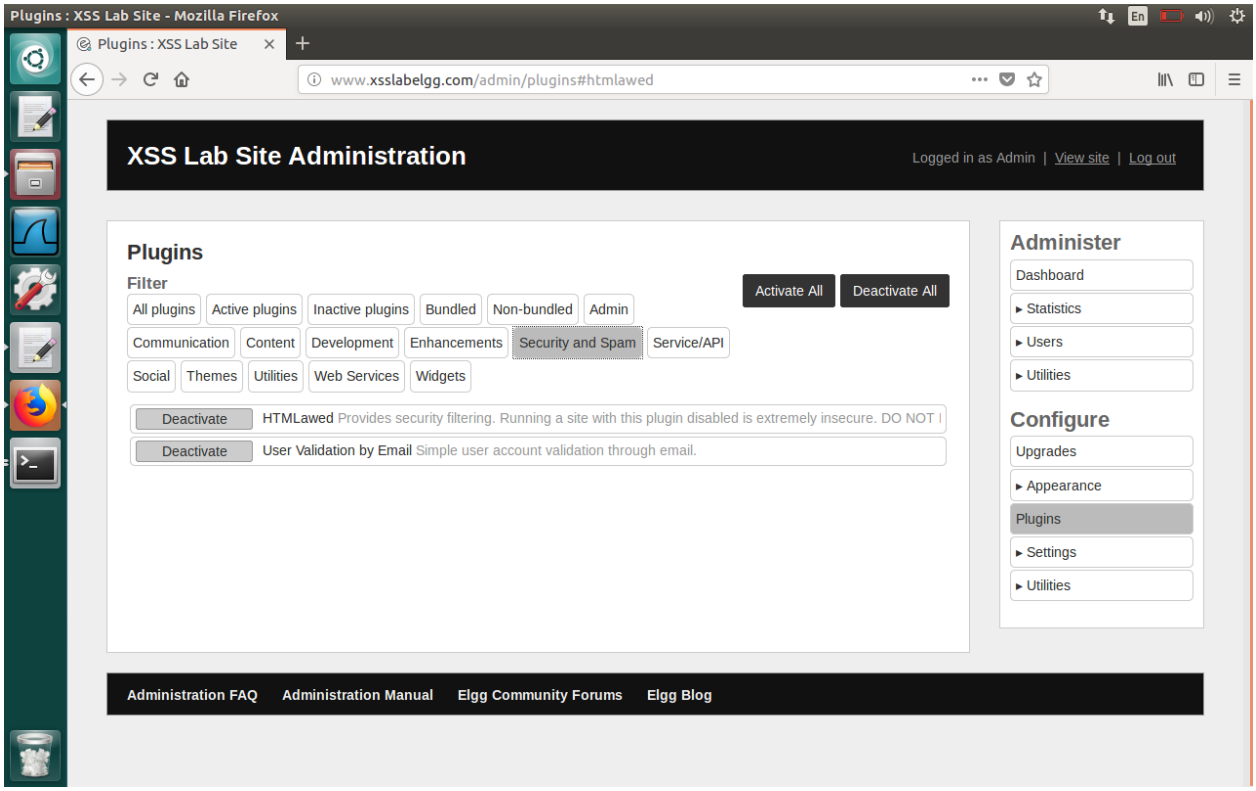
TASK 7:



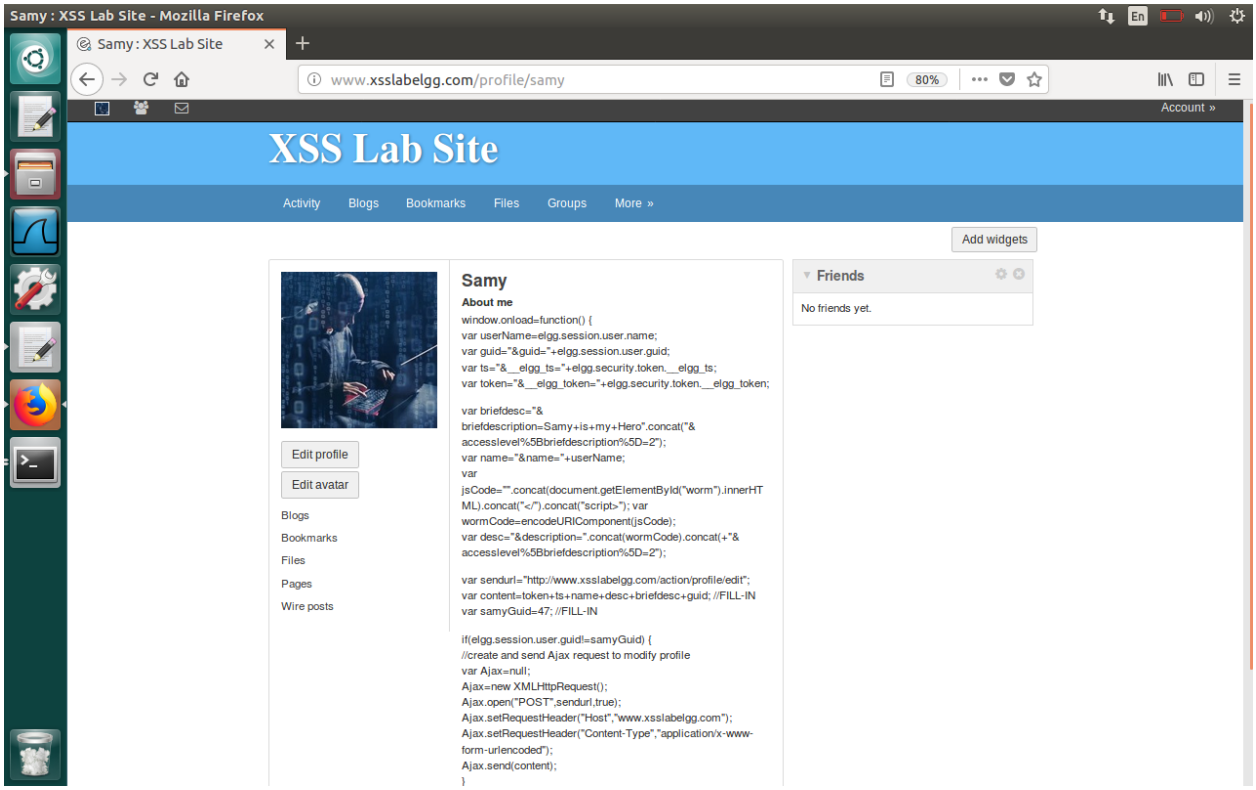
Screenshot 7.1: Adding the malicious code in Samy's description before countermeasures



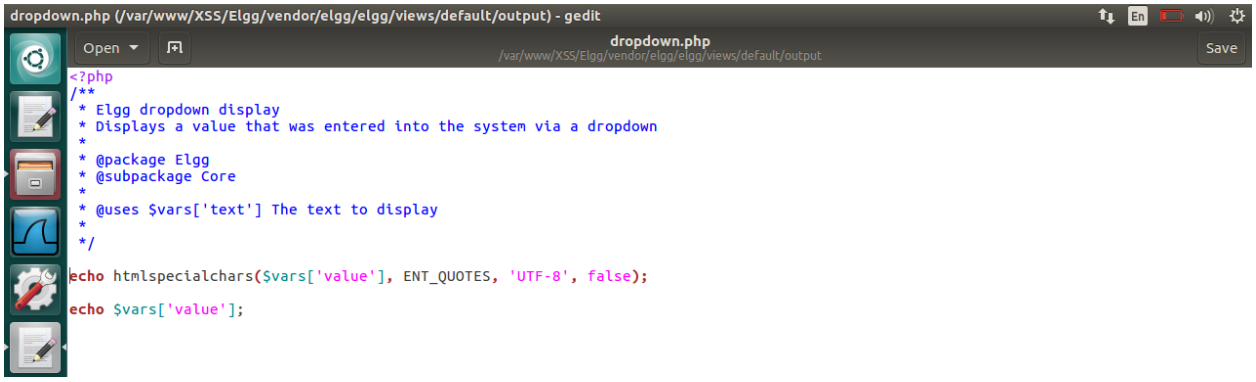
Screenshot 7.2: Samy's profile shows nothing in description since javascript code



Screenshot 7.3: plugin HTMLawed installed by admin



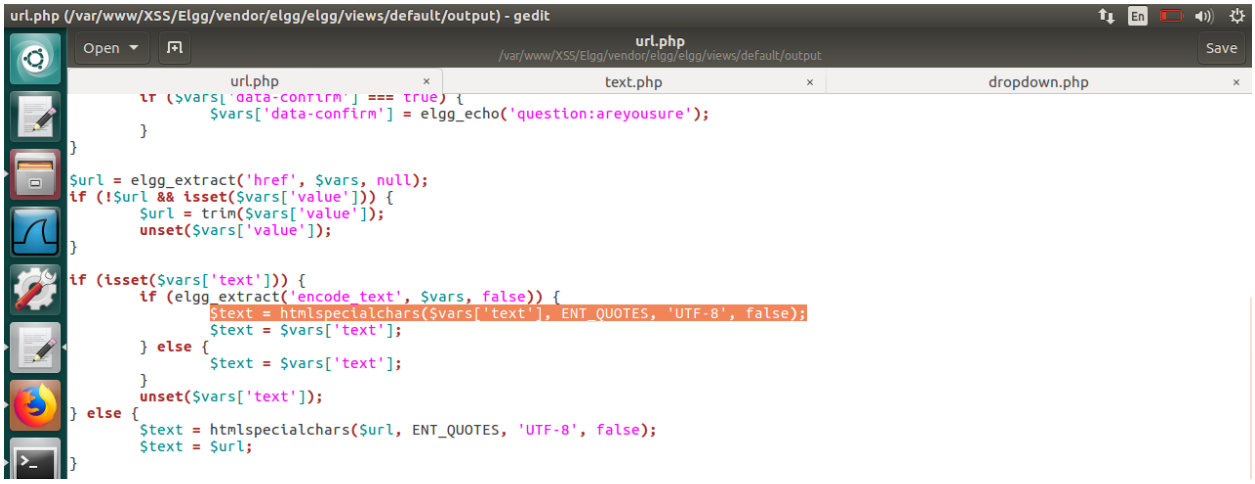
Screenshot 7.4: Samy's profile after HTMLawed installed



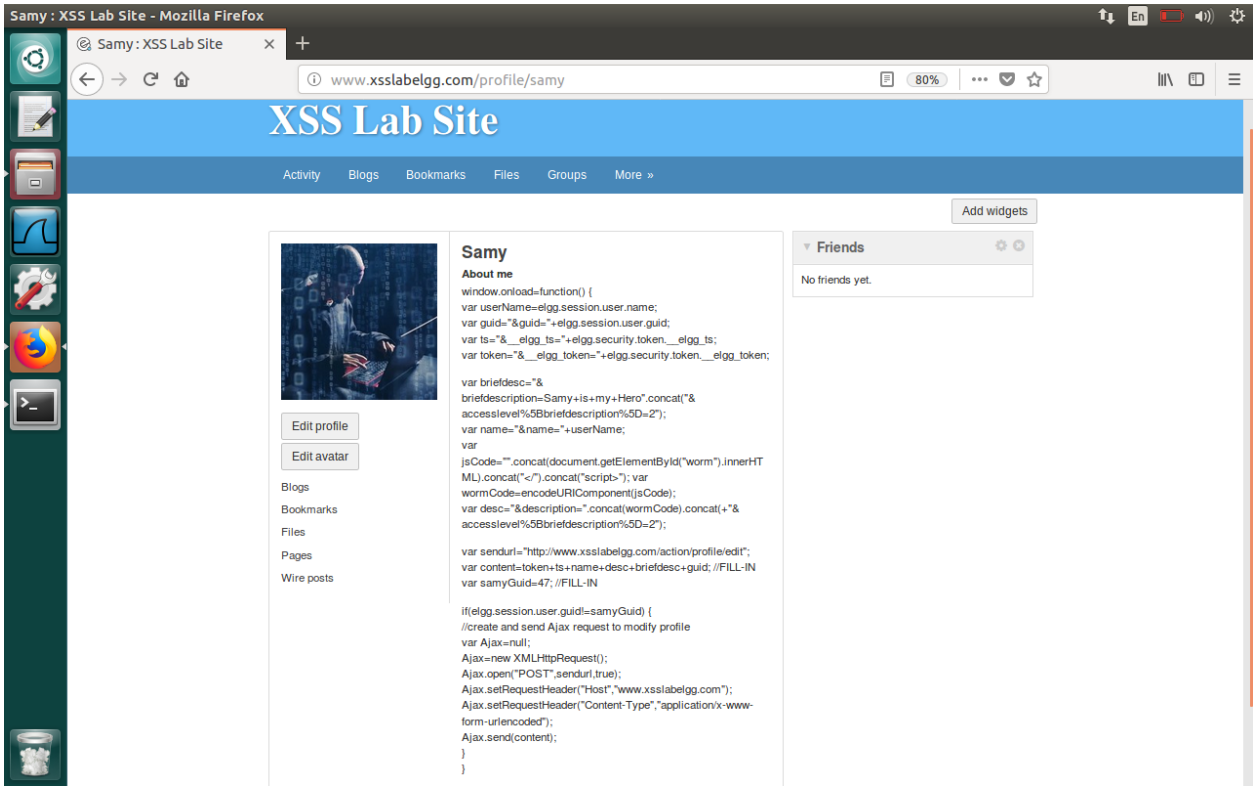
Screenshot 7.5: htmlspecialchars line in dropdown.php uncommented

Screenshot 7.6: htmlspecialchars line in email.php uncommented

Screenshot 7.7: htmlspecialchars line in text.php uncommented



Screenshot 7.8: htmlspecialchars line in url.php uncommented



Screenshot 7.9: Samy's profile after uncommenting htmlspecialchars line and enabling HTMLawed

There are two components in this task:

1. enabling HTMLawed plugin(done by admin of the site)
2. uncommenting HTML special characters

Enabling HTMLawed plugin:

On enabling this, the malicious code in the description of the attacker doesn't work. This is because the script tags in the code are removed as it can be seen in Screenshot 7.4. Since there are no script tags, the malicious code acts as a text instead of javascript code. Hence the attack is unsuccessful.

Uncommenting HTML special characters:

The lines with 'htmlspecialchars' are uncommented in the dropdown.php, email.php, text.php and url.php. The attack again fails. This line encodes the special characters in the description of the profile. The special characters like <,>,{,} are encoded hence there is no way the code will run.