Information Security: Cross-Site Scripting (XSS) Attack

Name: Chandan N Bhat

PES1201701593

Section H

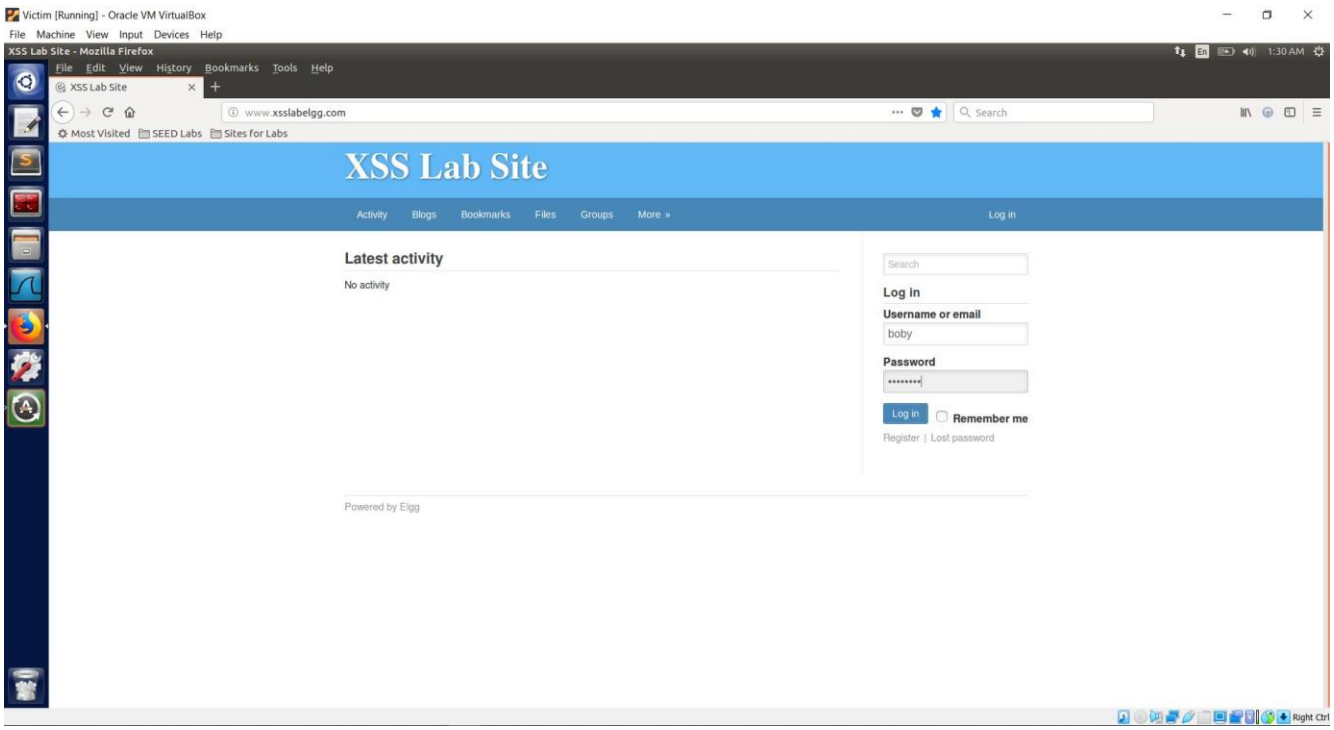
XSS is a very common security vulnerability found in web applications. It is a client-side code injection attack into web pages viewed by users. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application.

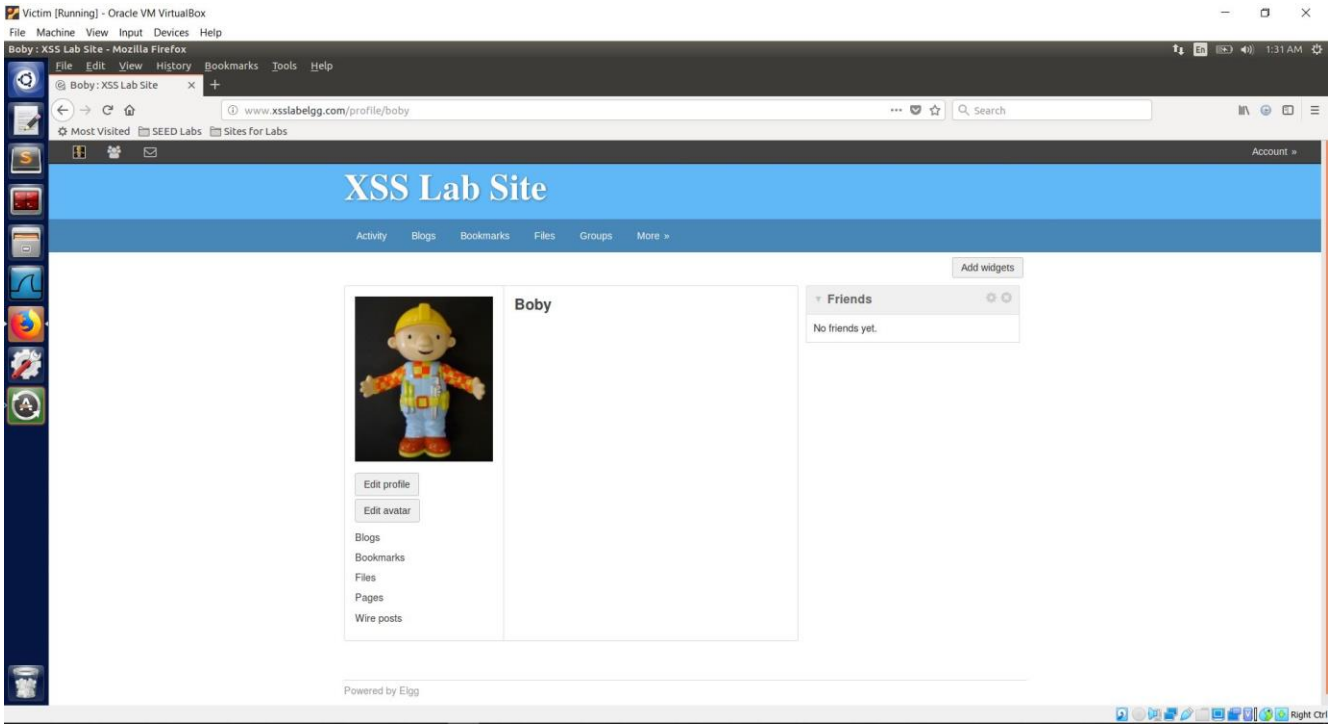
We will be using the 'Elgg' website www.xsslabelgg.com to demonstrate the XSS vulnerability.

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

In this task we will embed a Javascript program into our Elgg profile page. We will use the existing Boby's account as our account.

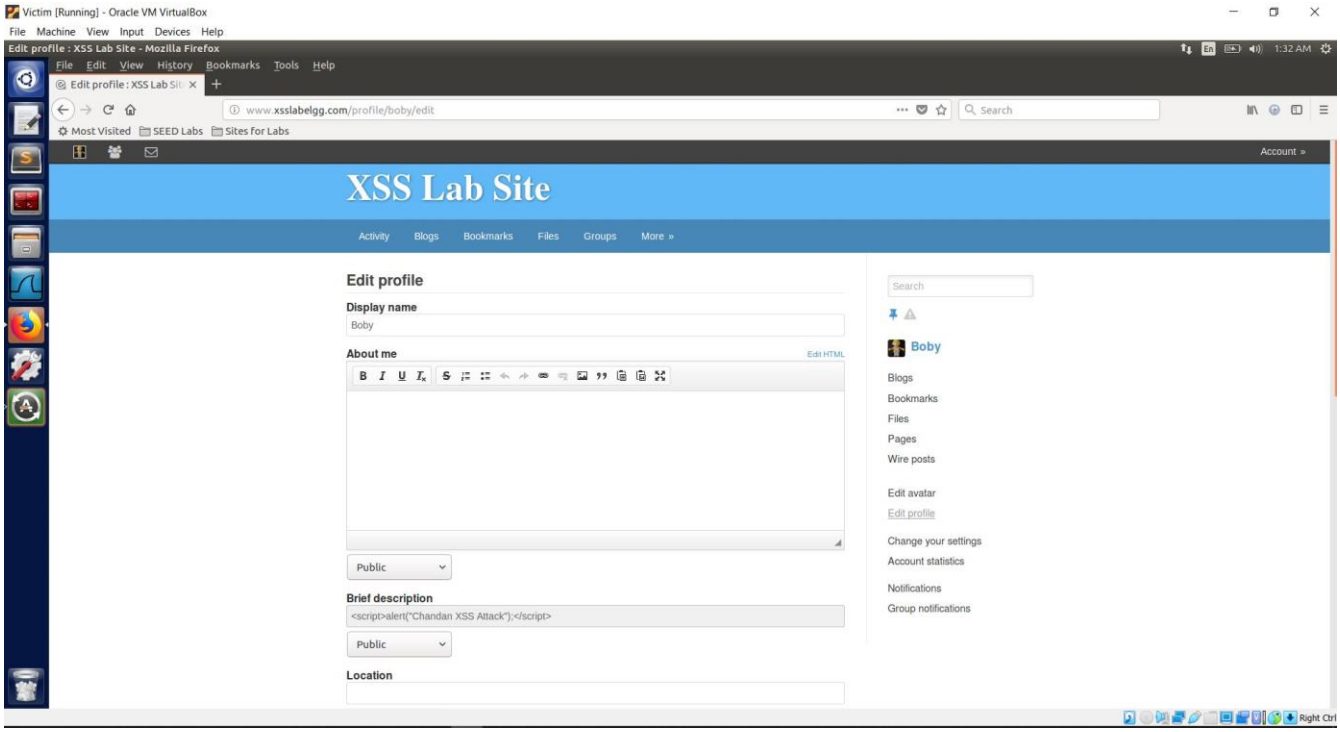
We login to the website www.xsslabelelgg.com as Boby as shown below. The username and password are "boby" and "seedboby" respectively.



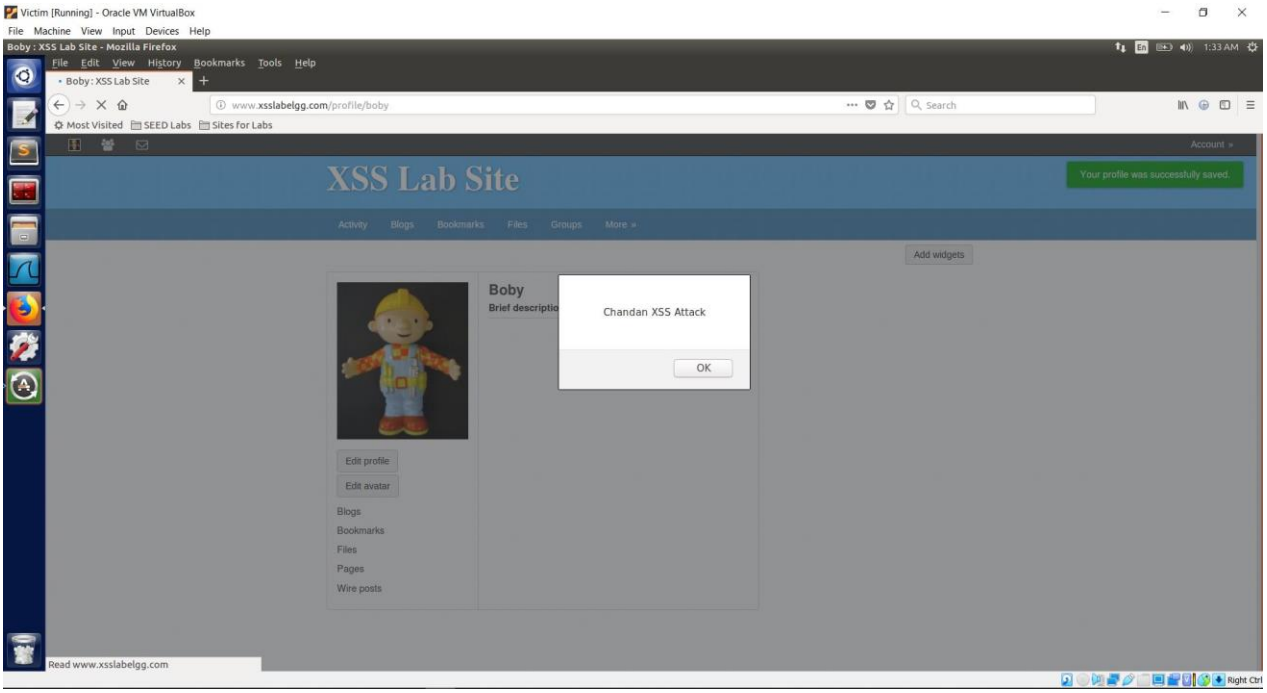


We will now update the profile and in the brief description field we will add the below Javascript code

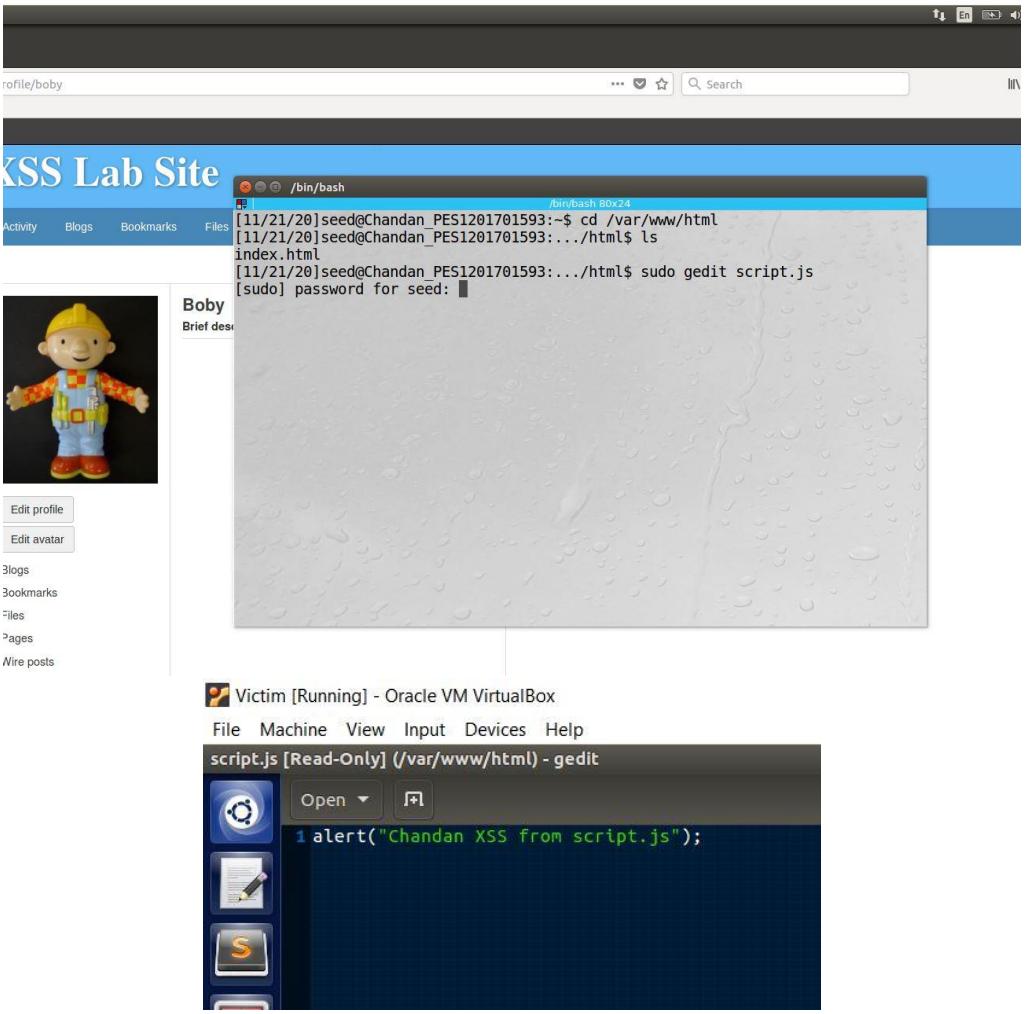
<script> alert("Chandan XSS Attack"); </script>



On submitting the form, we observe that we get an alert in the webpage displaying "Chandan XSS Attack". Thus, this indicates that the Javascript code that we embedded was executed. We can run multiple Javascript statements separated by ';'.

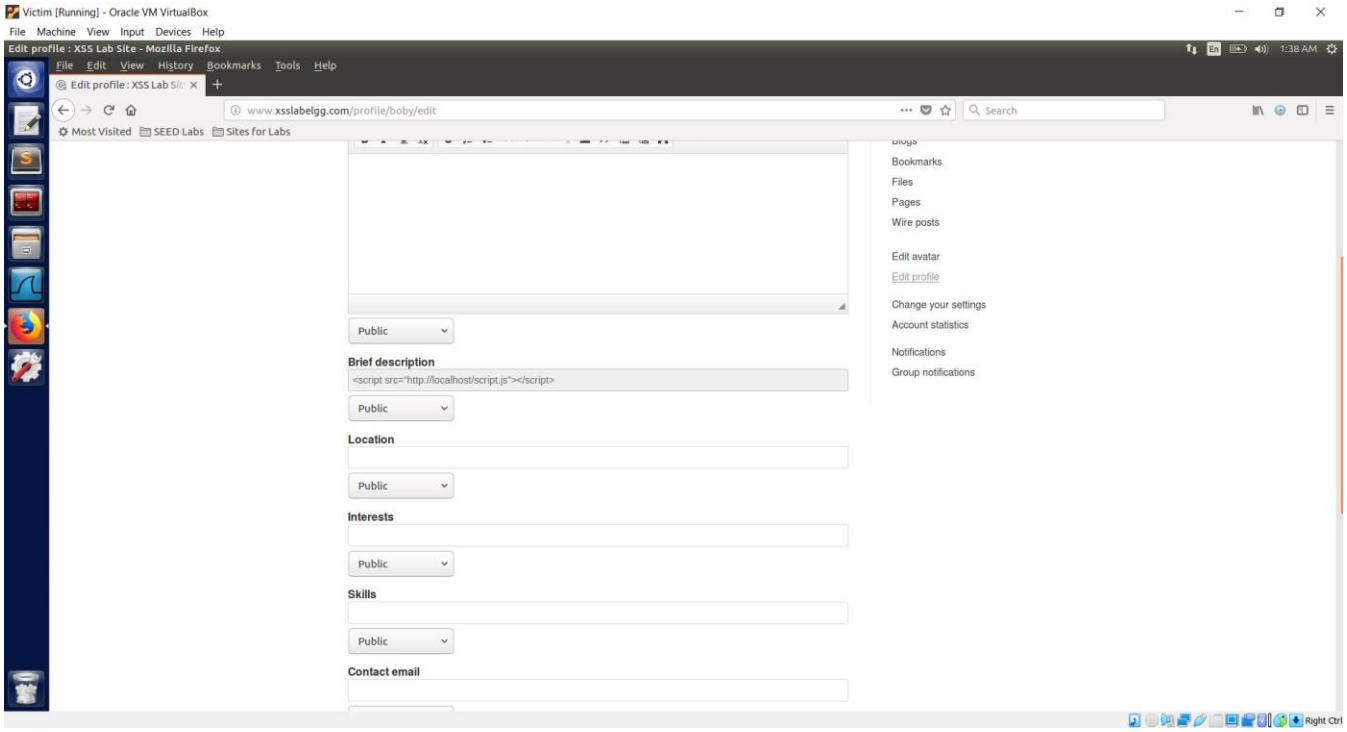


If the Javascript code to be executed is very long, then we can store it in a " .js " file and reference it using the 'src' attribute in the script tag. This is demonstrated in the below screenshot. To make the script.js file accessible we create it in "/var/www/html/ " directory.

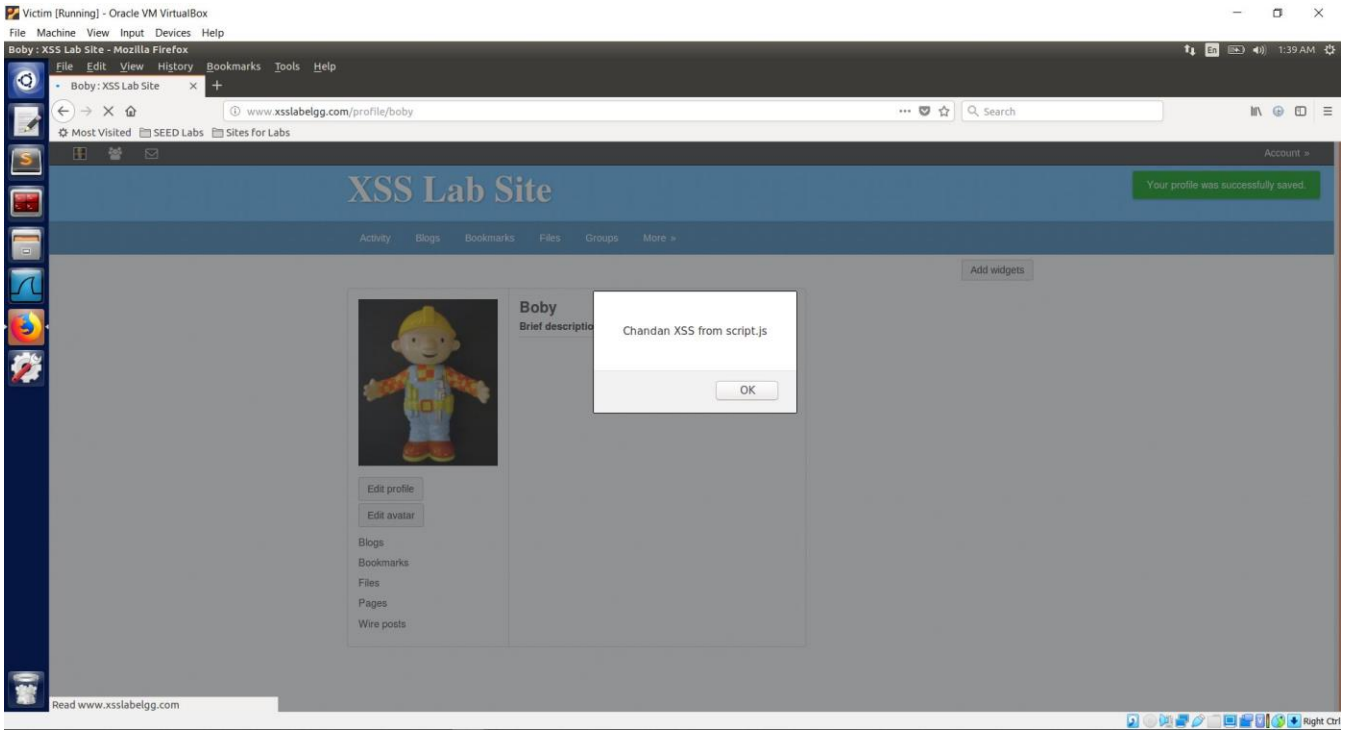


We again embed a script tag in the form field, now referencing the script.js file and save the profile as shown below.

<script src="http://localhost/script.js"> </script>

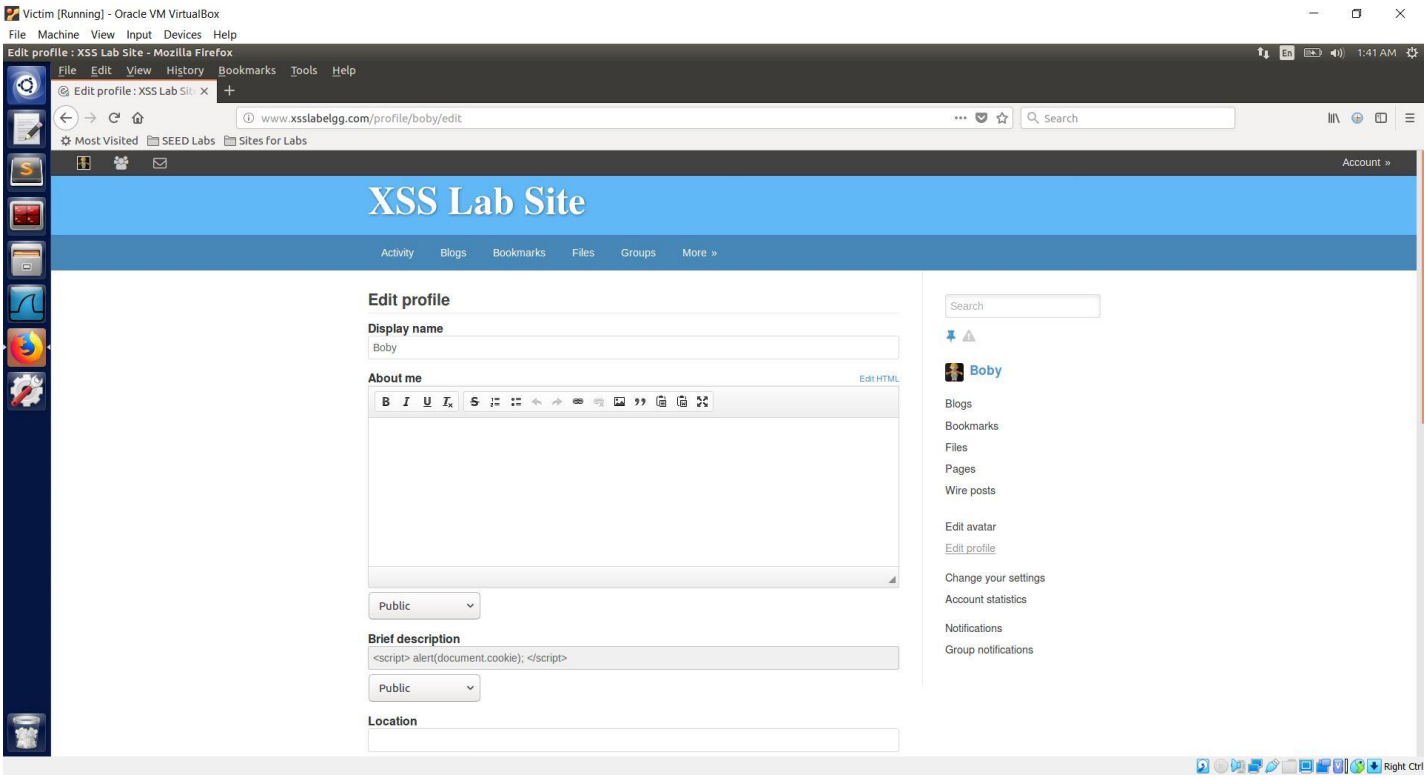


From the below screenshot we observe that the Javascript code in the script.js file is successfully executed.

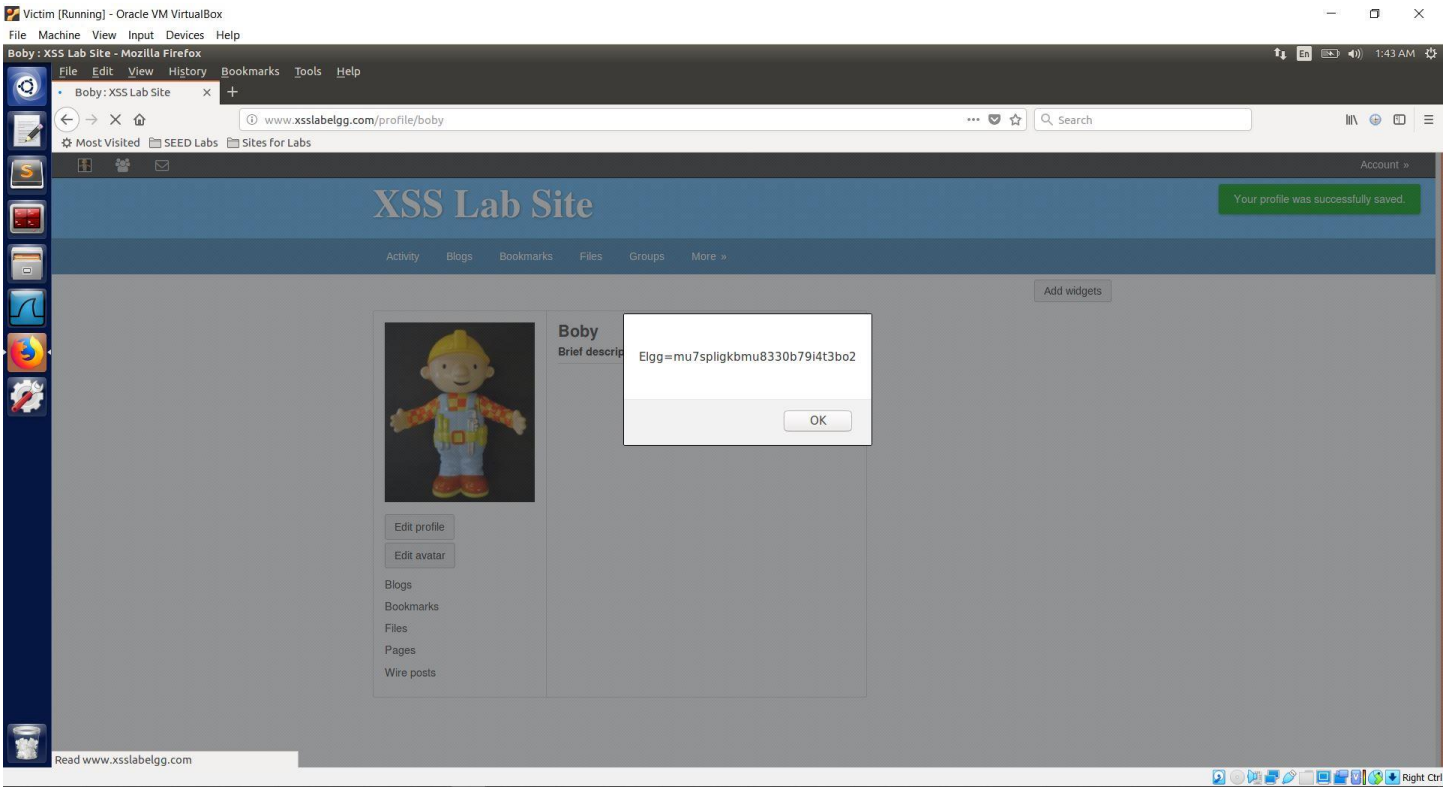


Task 2: Posting a Malicious Message to Display Cookies

Next we will try to display the user's cookie by embedding Javascript code. We make use of "document.cookie" to display the user's cookie as shown below.



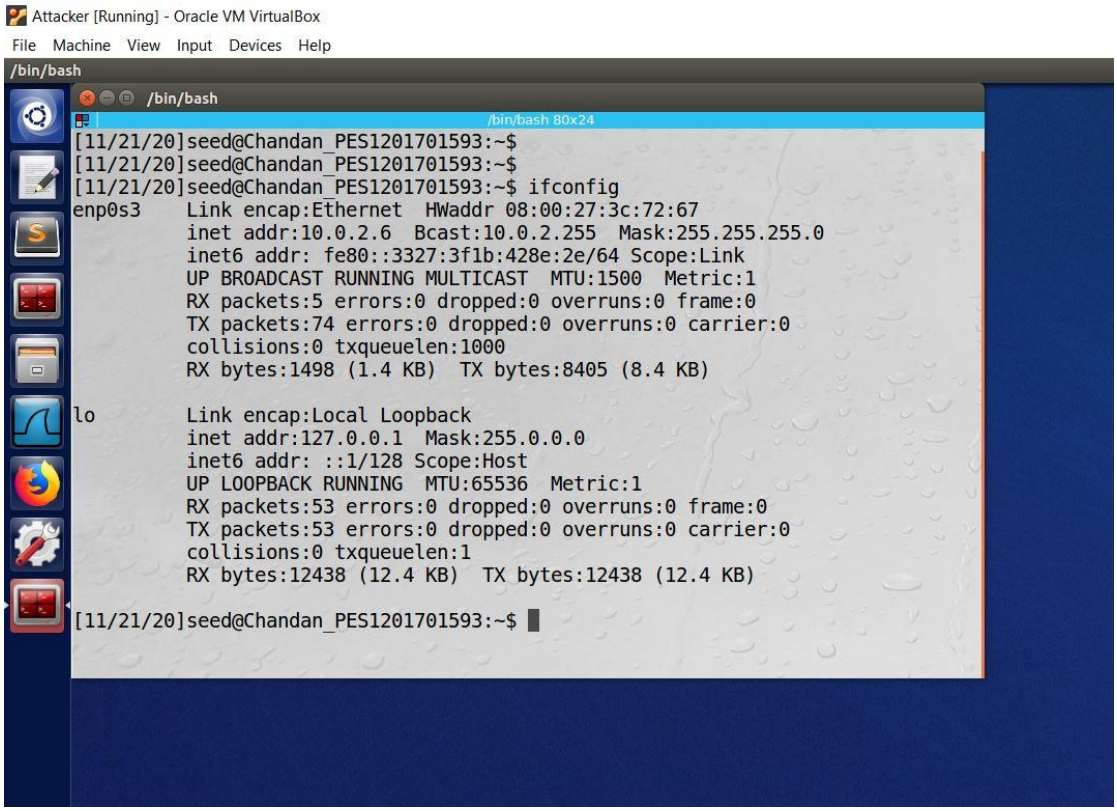
On saving the profile we observe that we get an alert displaying the user's cookie as shown below.

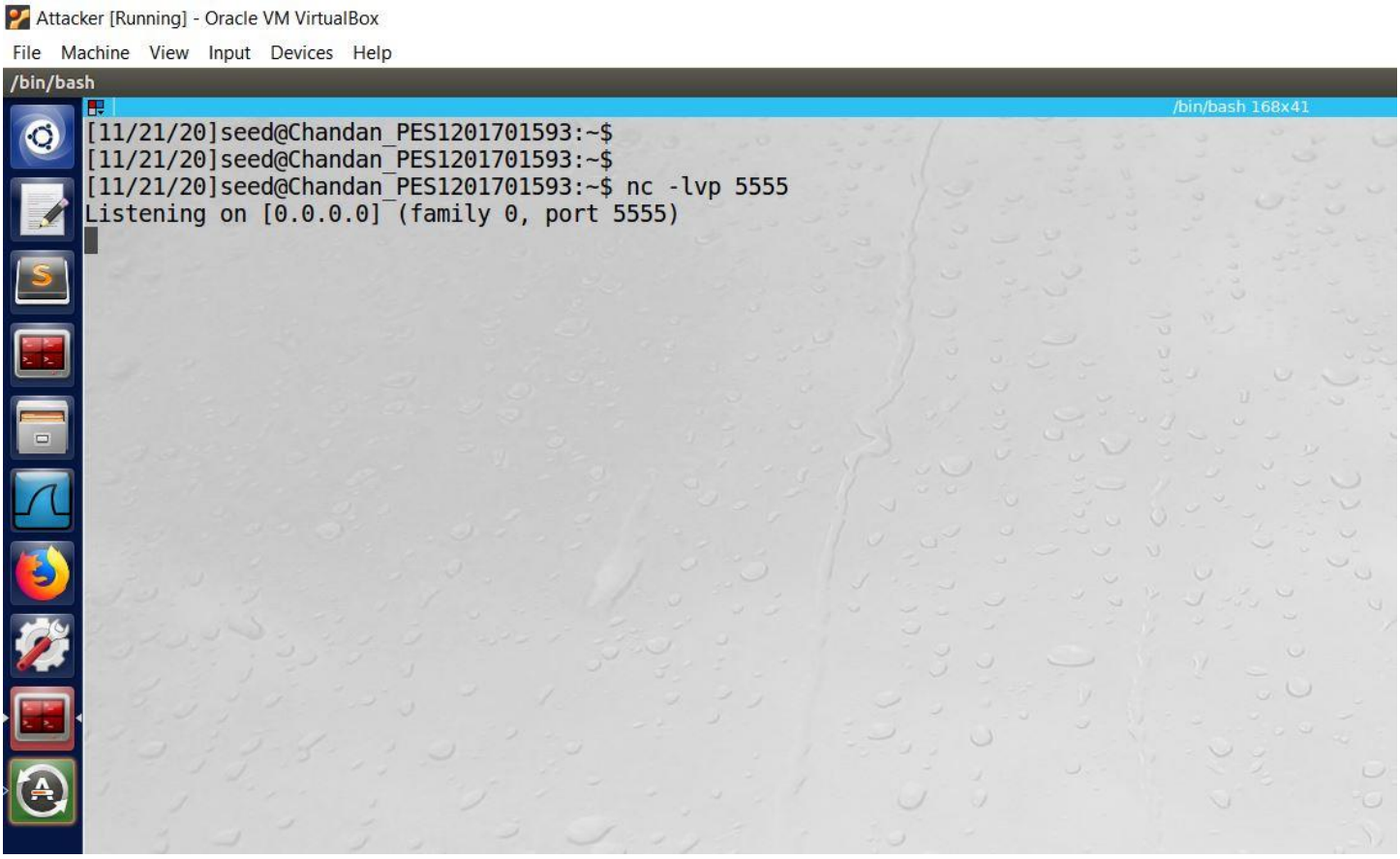


Thus we successfully embedded a Javascript code that displayed the user's cookie.

Task 3: Stealing Cookies from Victim's Machine

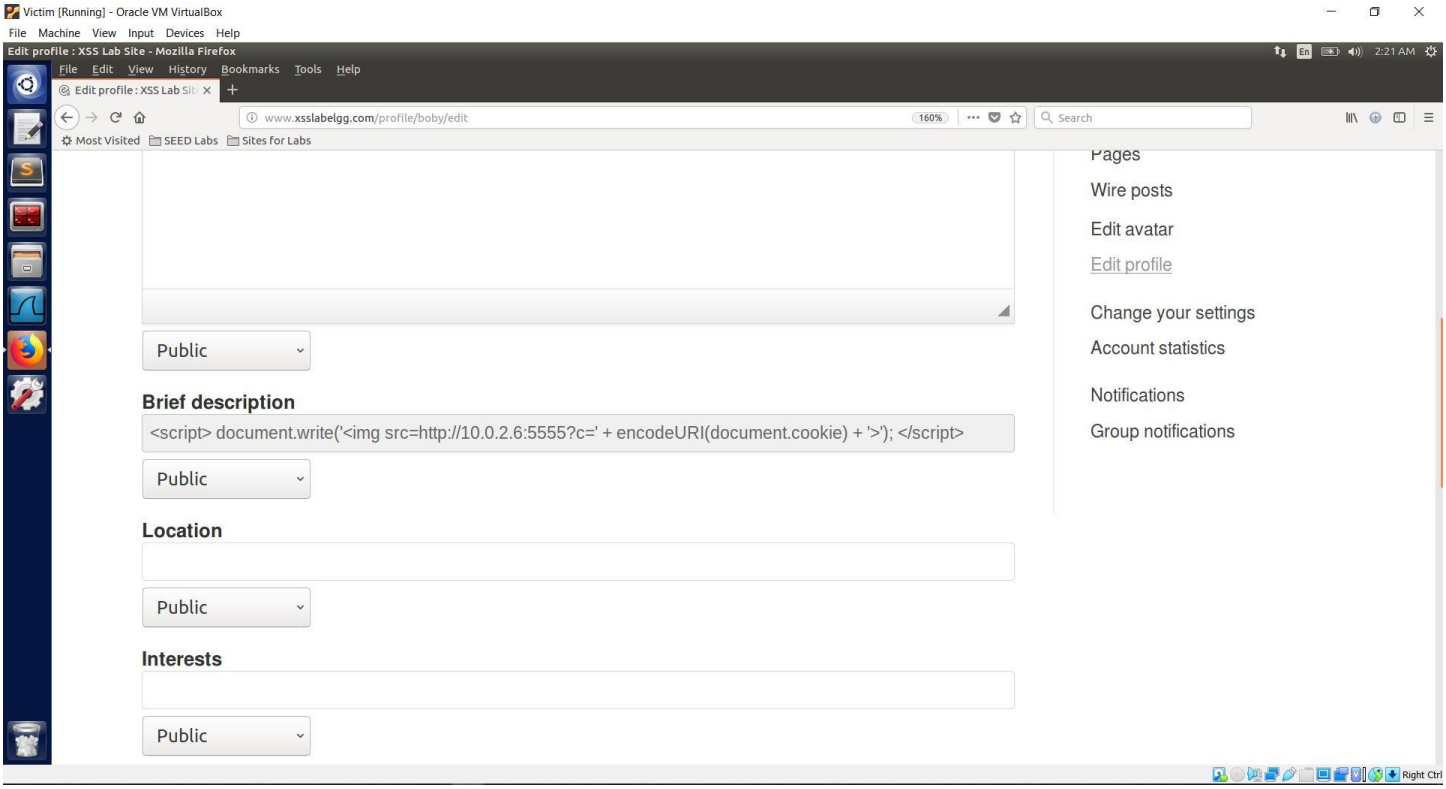
Now we will try to retrieve the cookie of a user than just displaying it as an alert. We will set up a listener at port 5555 on Attacker's machine (10.0.2.6) as shown below.



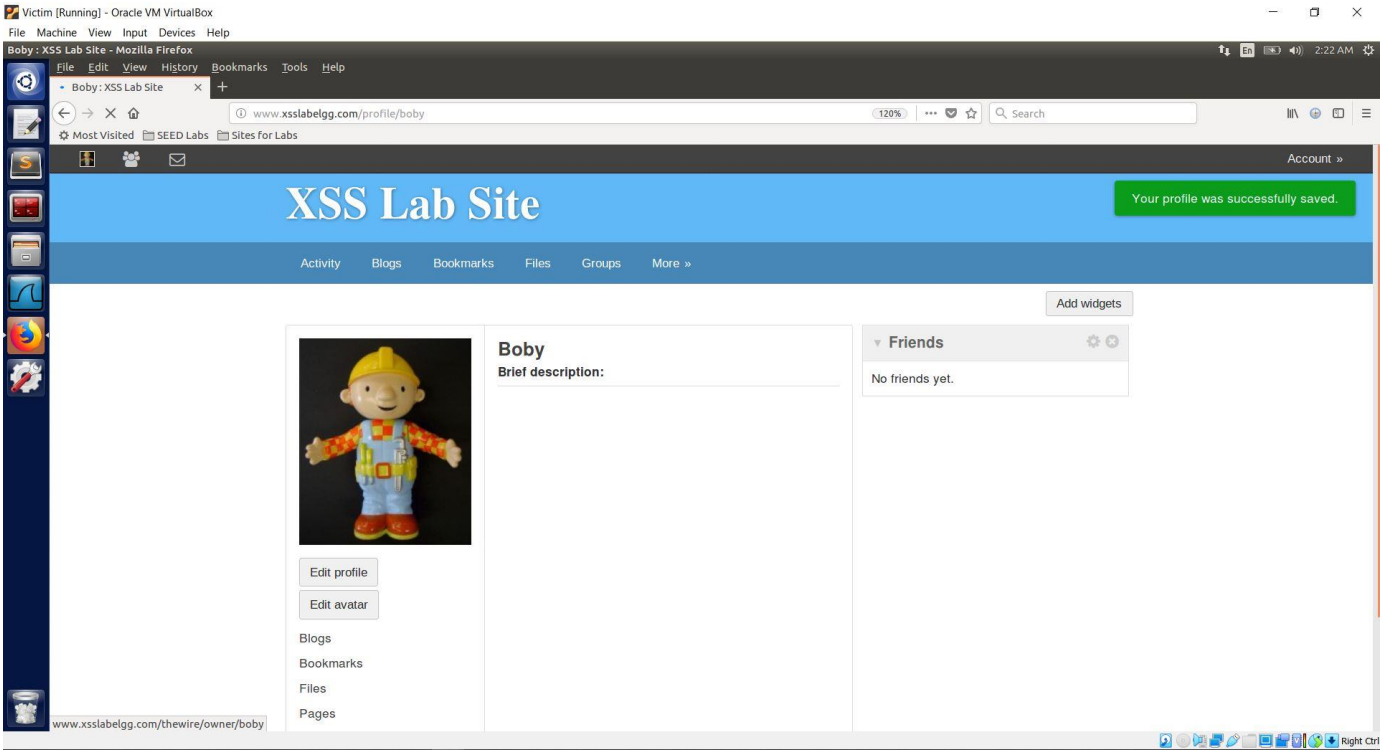


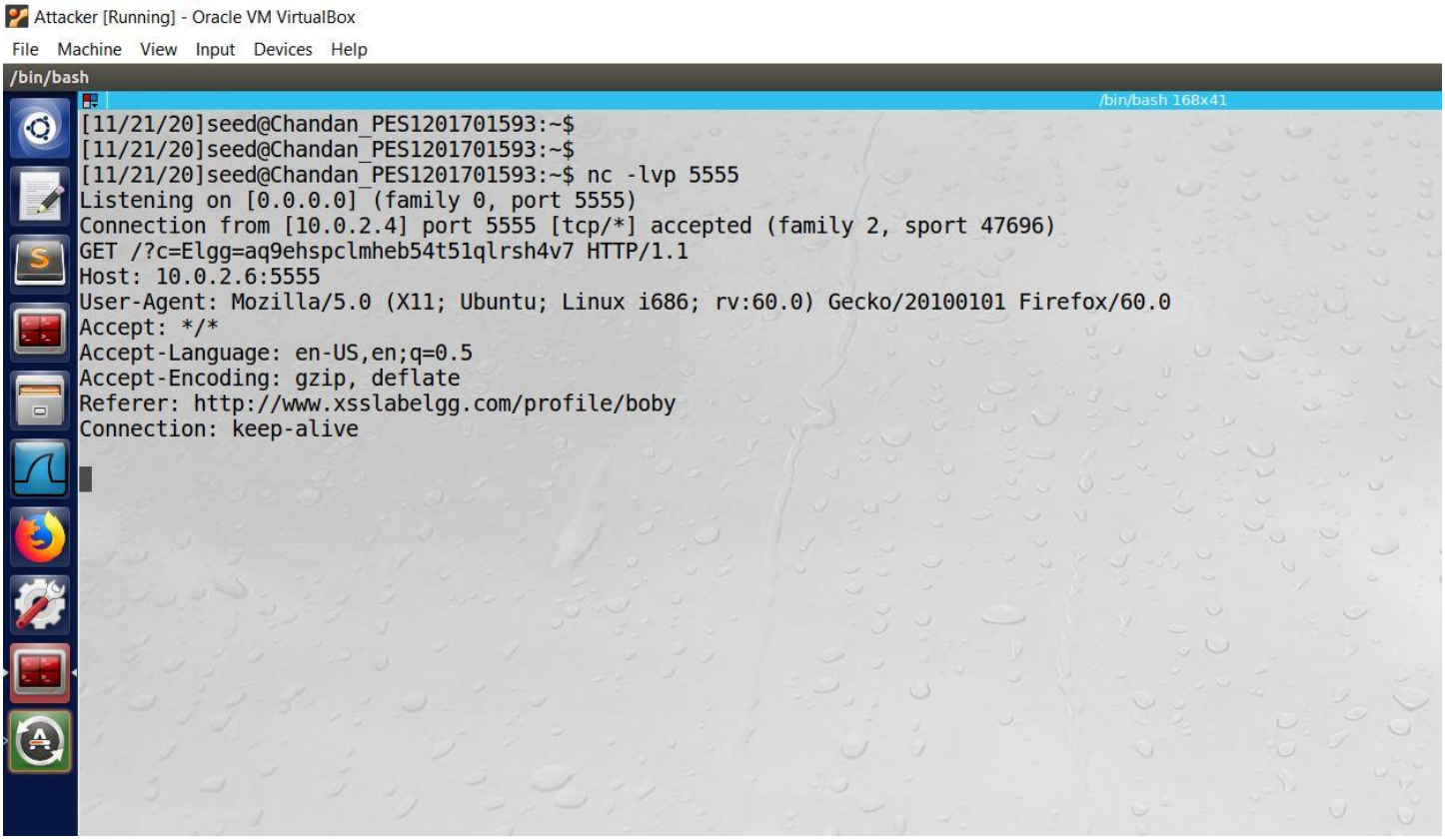
To receive a request from the Elgg website to the listener on the attacker machine, we can make an <img> tag with the source attribute set to the IP address of the attacker machine with the cookie as a part of the query string as the img tag makes a GET request to the url. We can use the document.write() method in Javascript to achieve this, as shown below.

The XSS script is typed into the brief description field.



On saving the profile we observe nothing happens on the webpage unlike earlier. But a GET request is made to the attacker machine listening on port 5555.

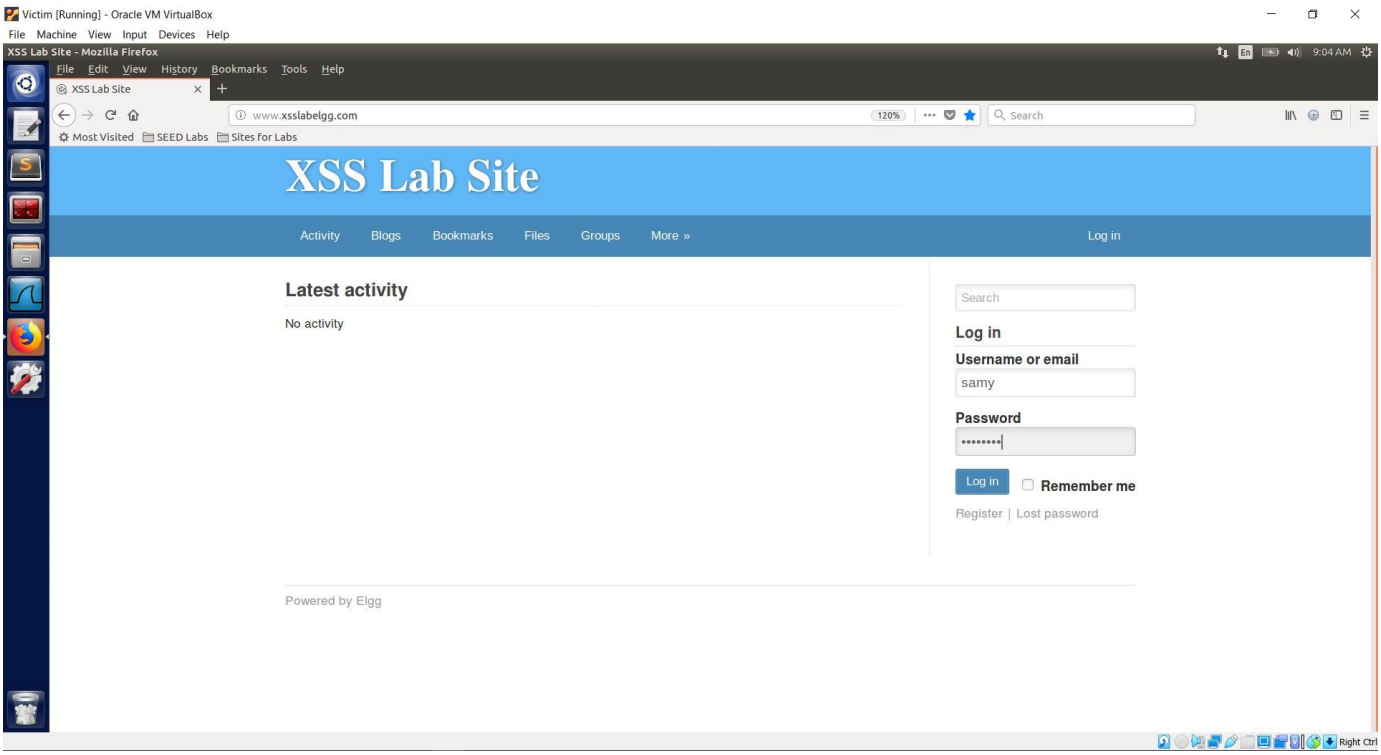


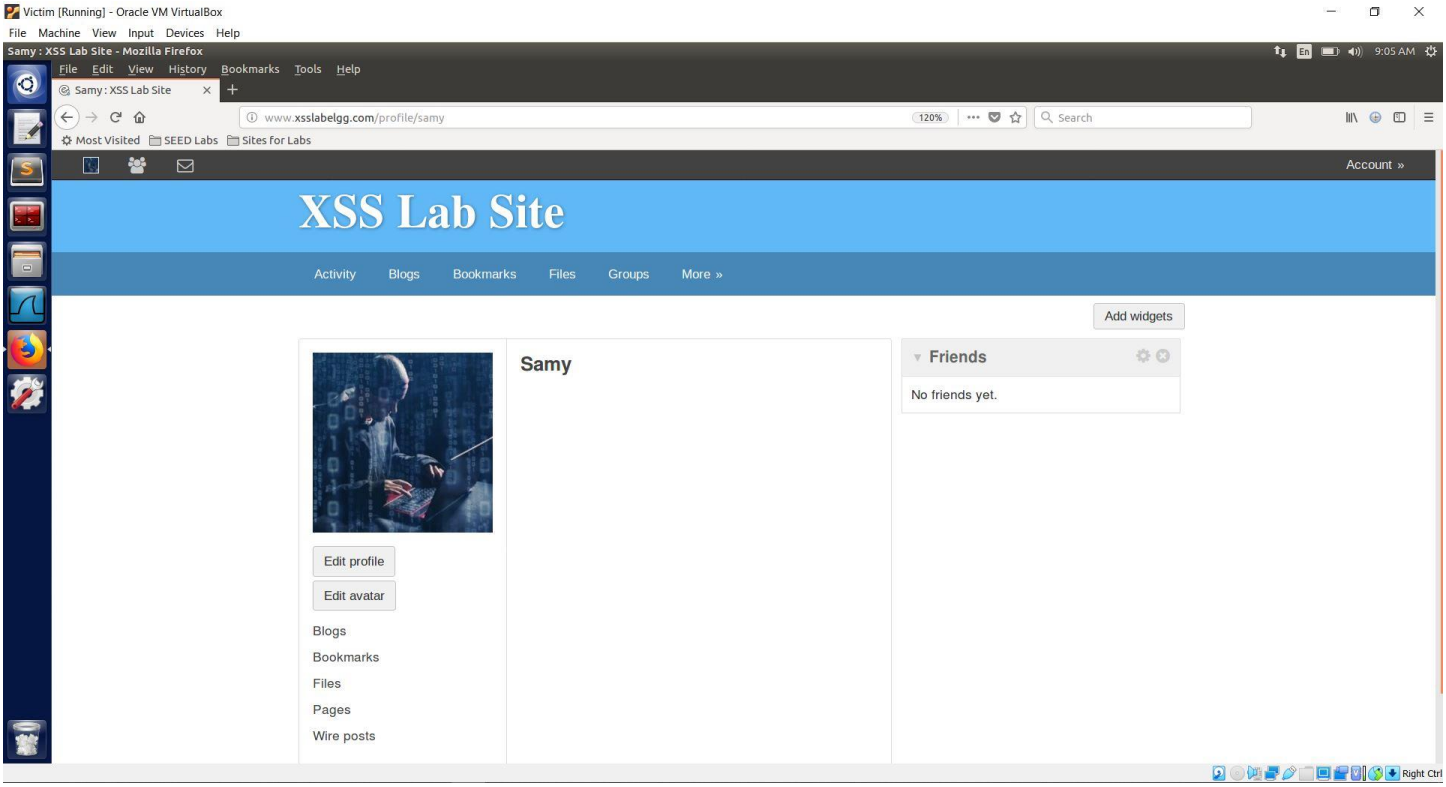


The above screenshot shows that the GET request has the user cookie as a part of the query string.

Task 4: Becoming Victim's Friend

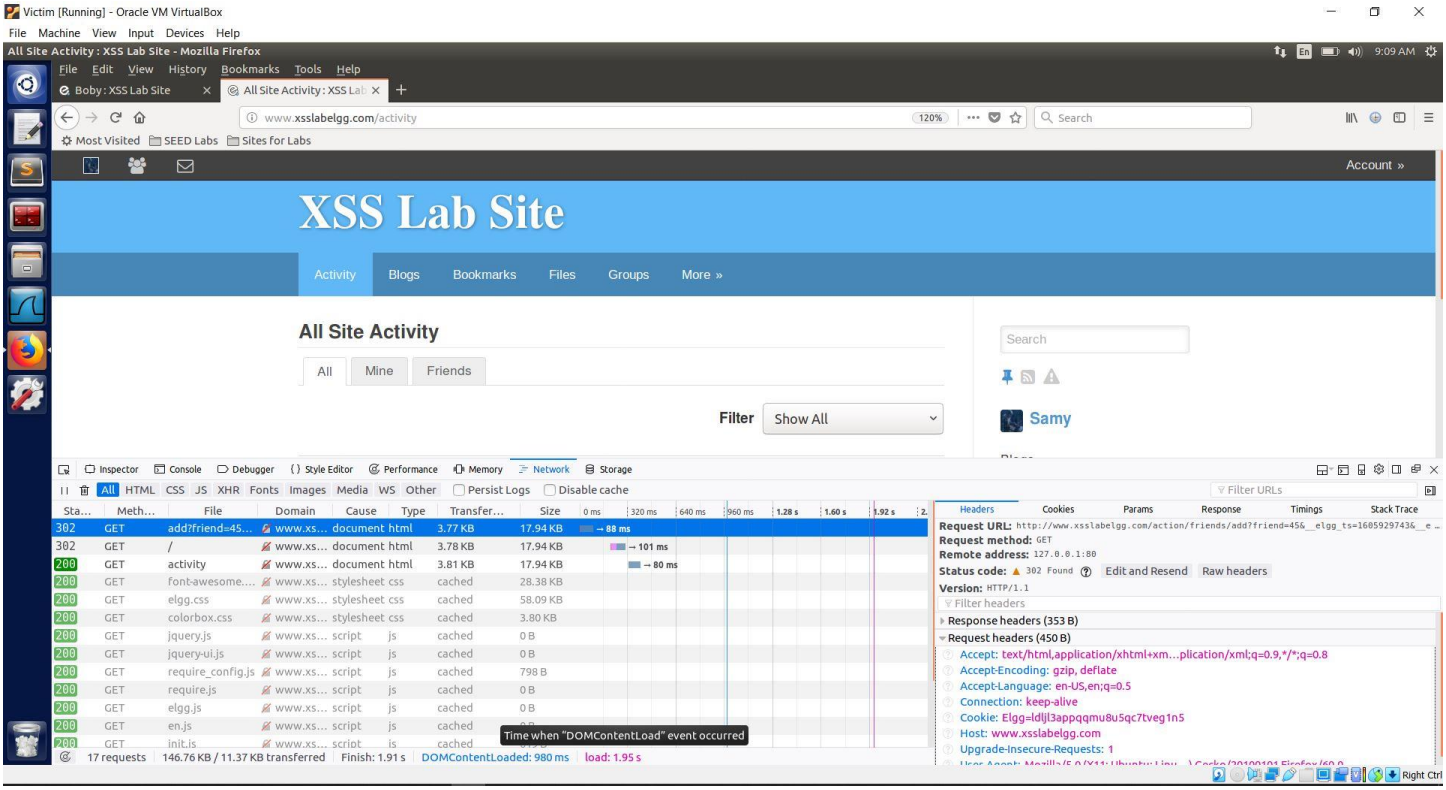
Now we will login as user Samy and try to add Samy as a friend to all those users who visit the profile of Samy.



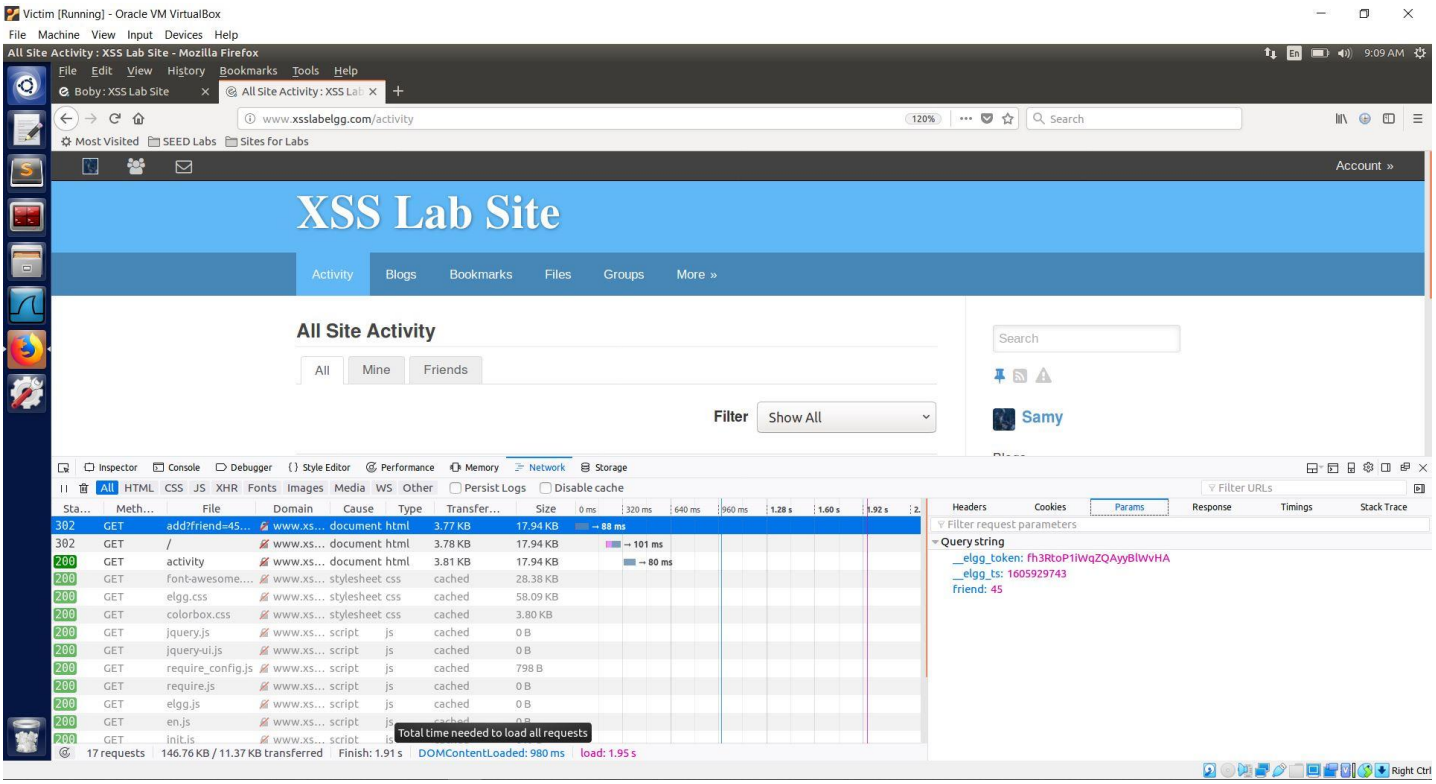


First, we need to see what request is sent when a user adds another user as a friend.

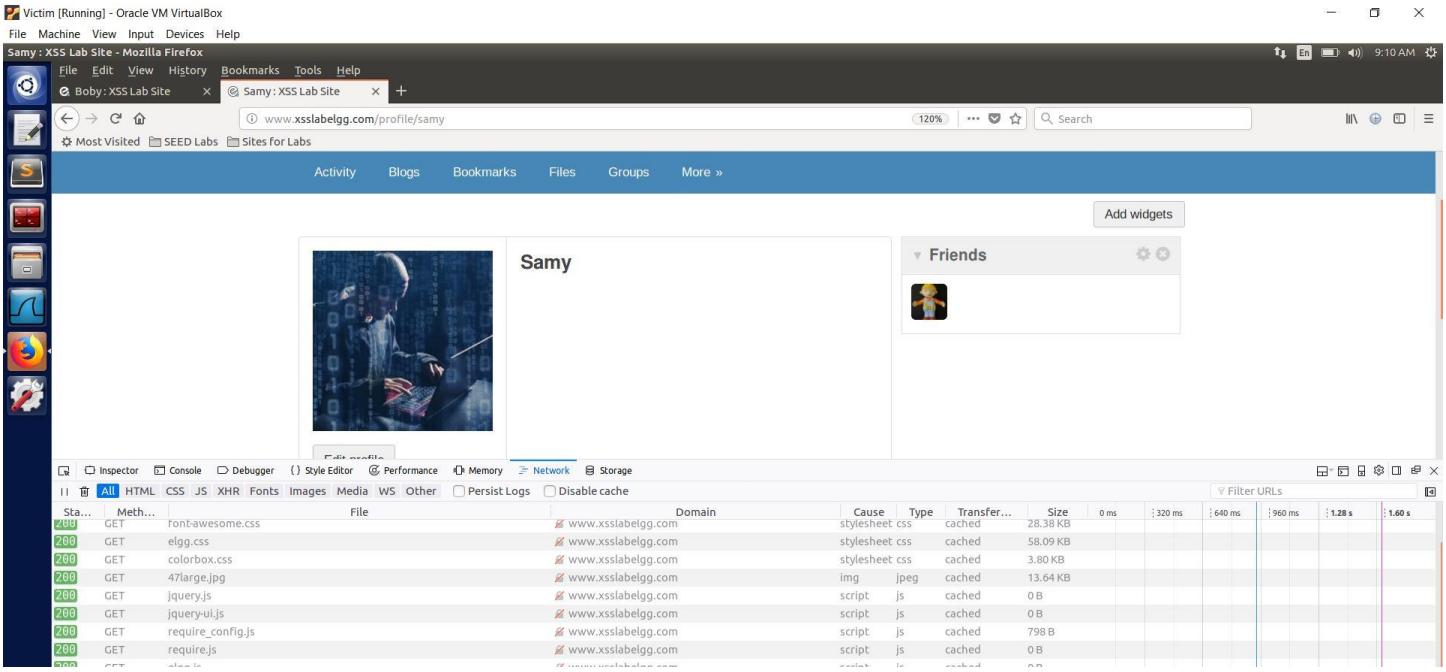
Using the developer tools, we will add Boby as our friend and observe the request and parameters sent.



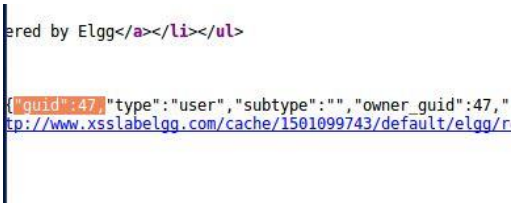
We observe that on adding Boby as a friend, a GET request is sent to www.xsslabelgg.com/action/friends/add?friend=45 where 45 indicates the id of the user, 45 being Boby's Id. The below screenshot shows the parameters sent as query string with the GET request.

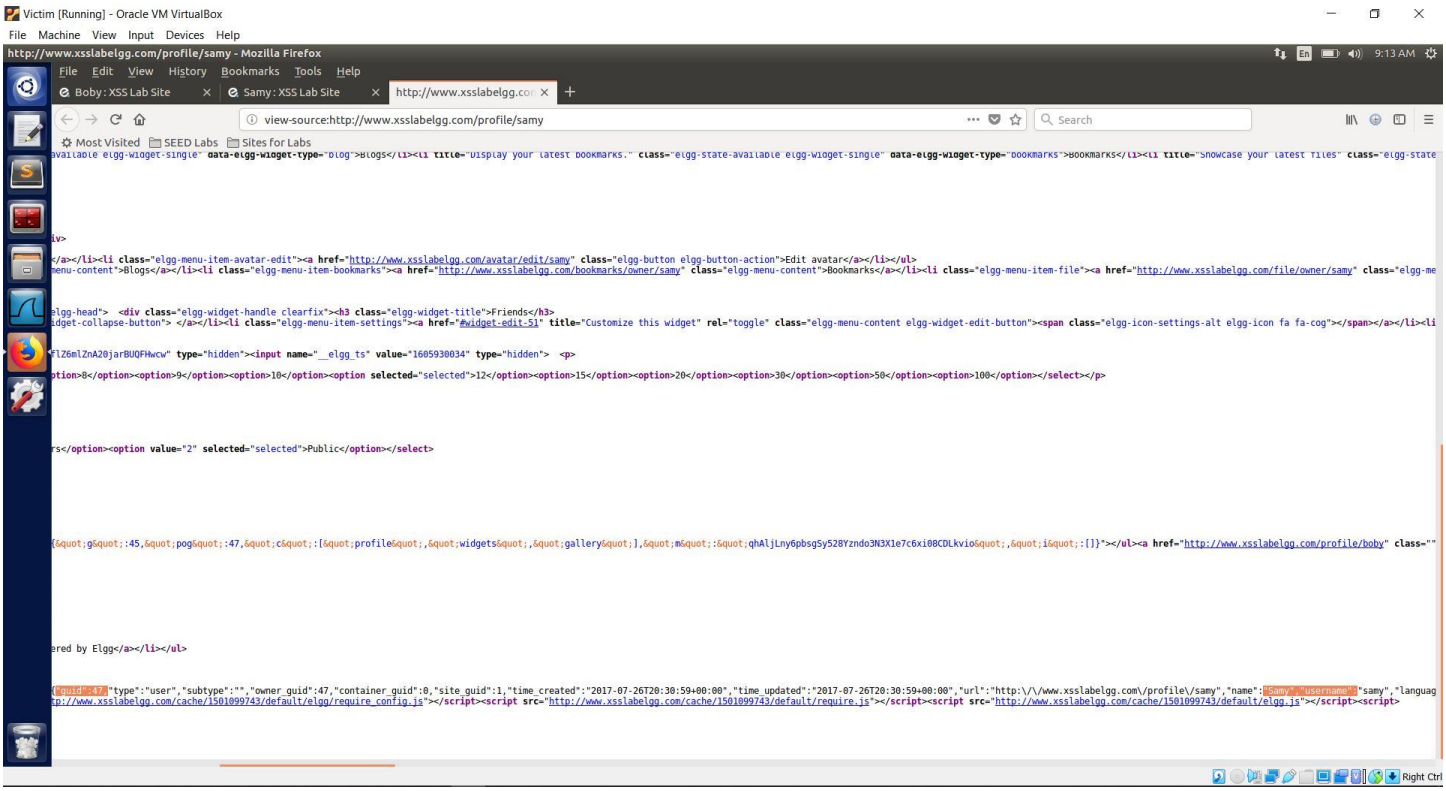


We observe that Boby is added as Samy's Friend.



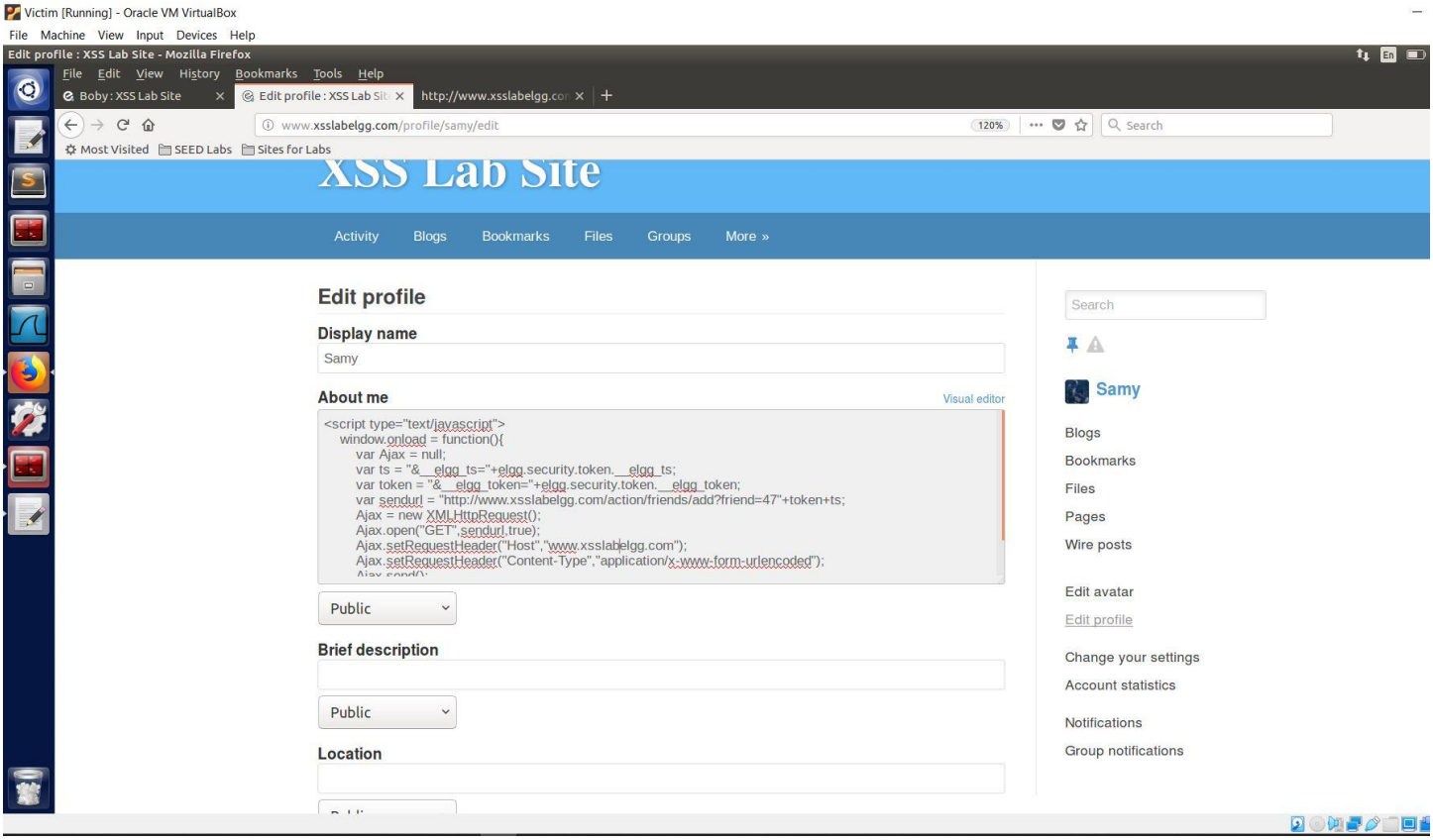
Since we want to achieve adding Samy as friend of all users who visit Samy's profile, we need to find out Samy's Id(guid). This can be found from the page source as it is stored in a Javascript variable as shown in the below screenshot.



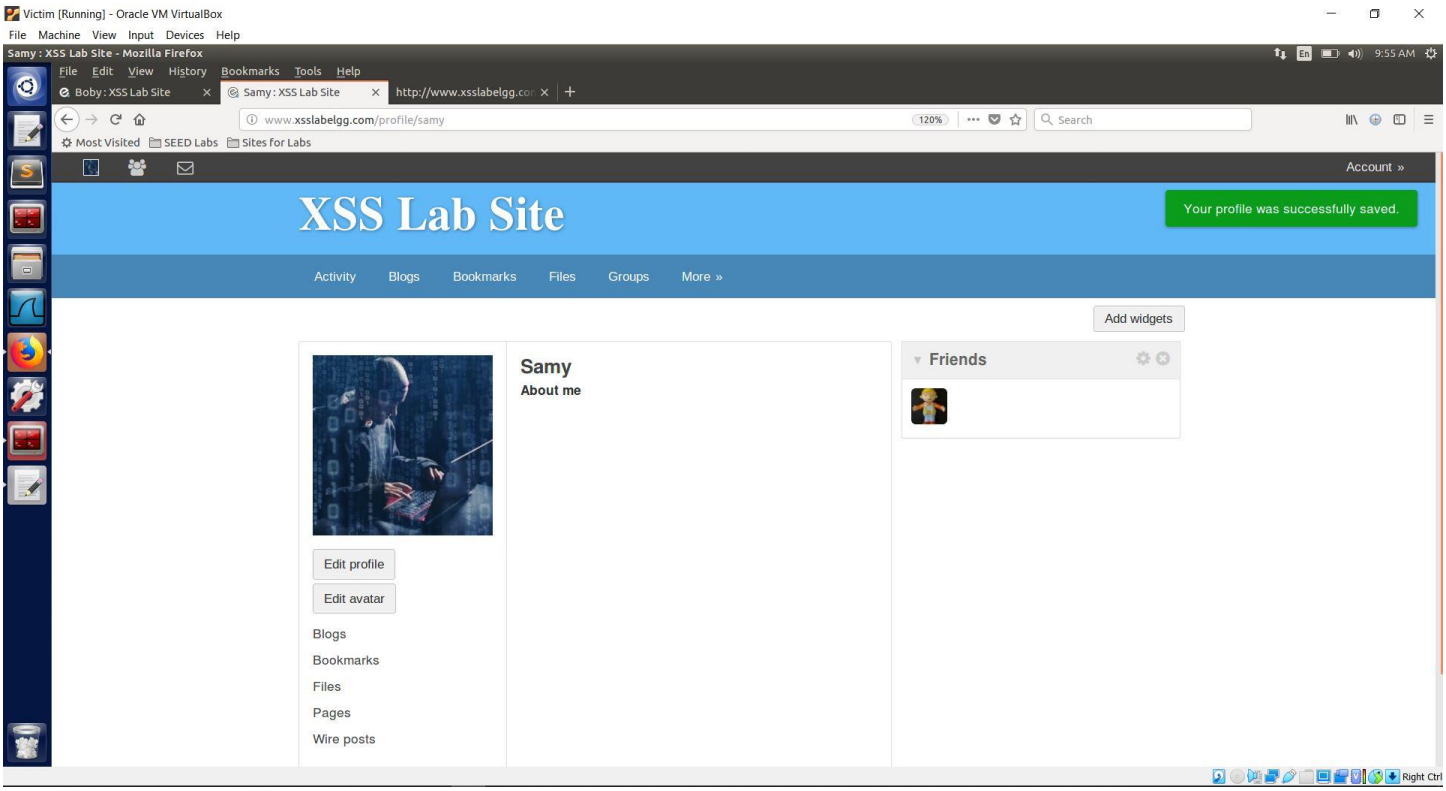


We observe that the guid of Samy is 47. Thus a valid request to www.xsslabelgg.com/action/friends/add?friend=47 adds Samy as the user's friend.

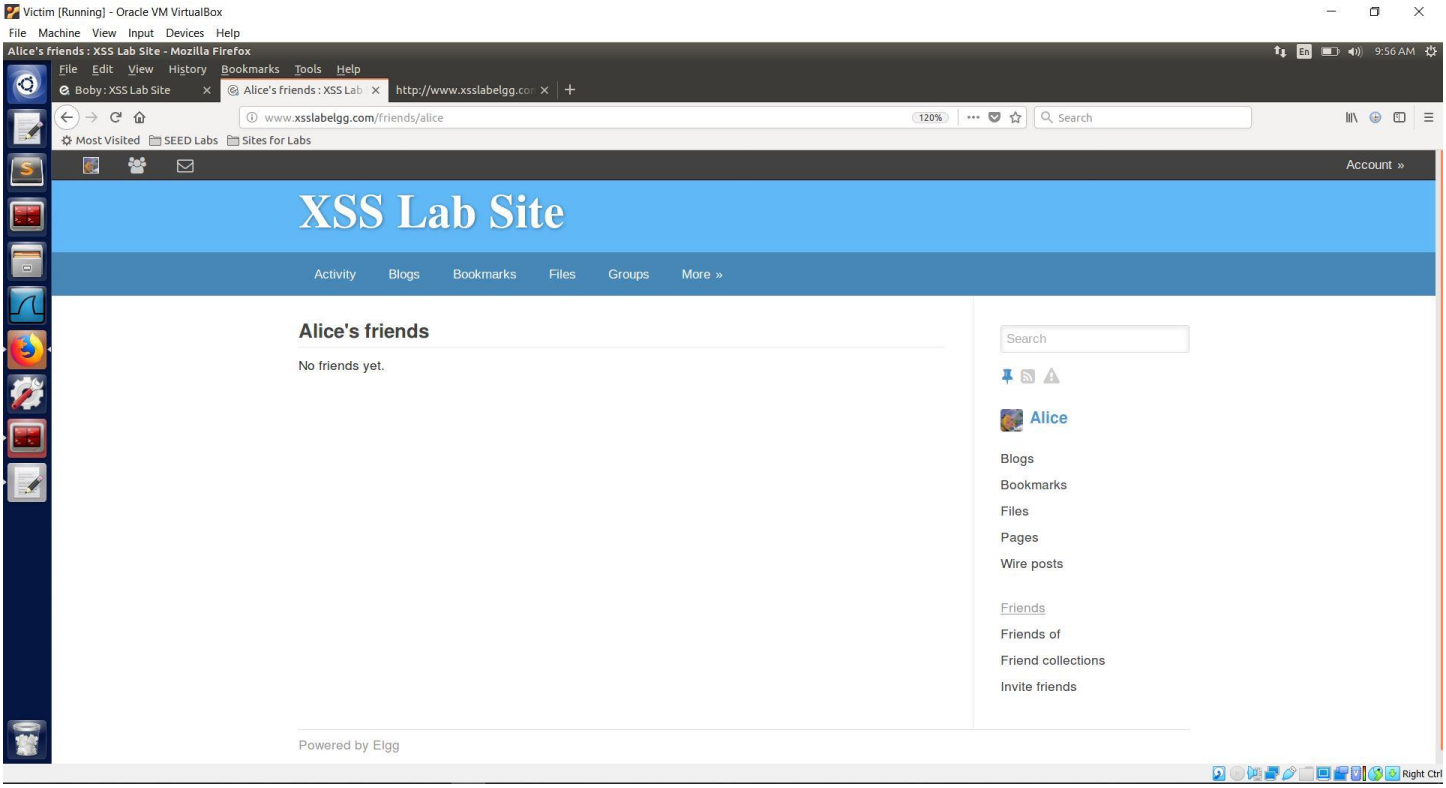
The script that achieves this is shown below.



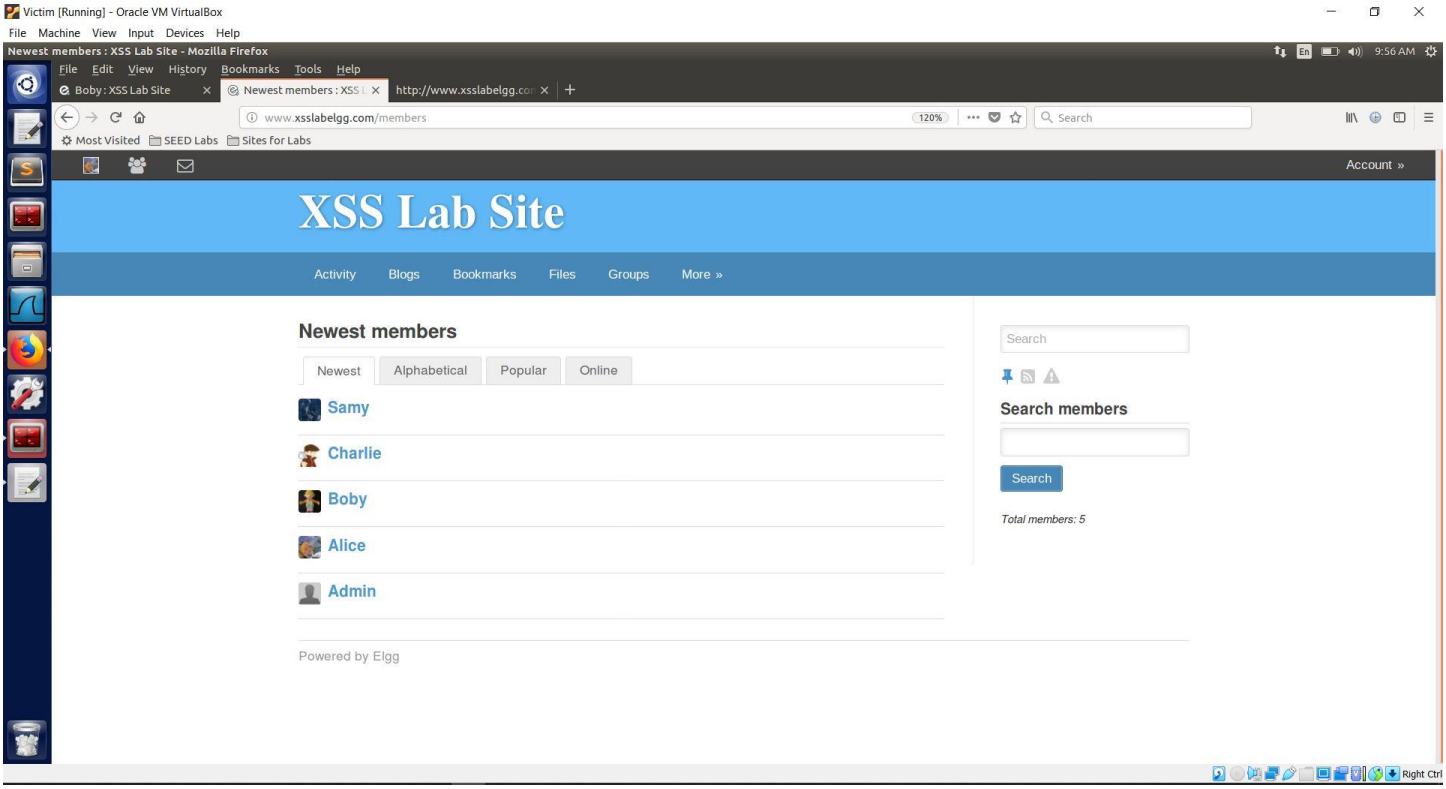
Using Ajax XMLHttpRequest we send a GET request to the above mentioned URL along with the csrf tokens to add Samy as the user's friend. On saving the above form we observe that nothing appears on the webpage, just a popup indicating that the profile was updated.

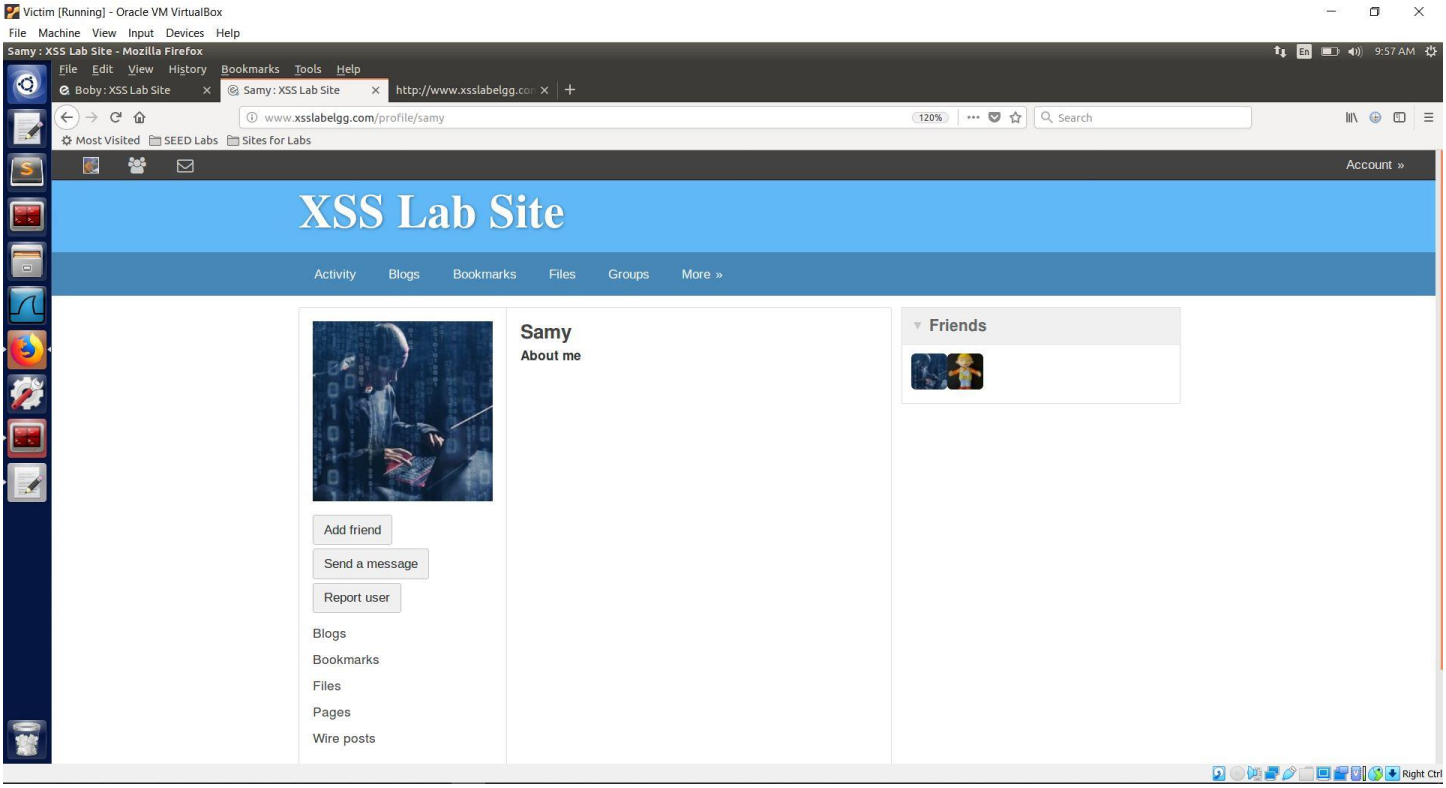


Now we login to the website as Alice, initially we observe that Alice has no friends in the list as shown in the below screenshot.



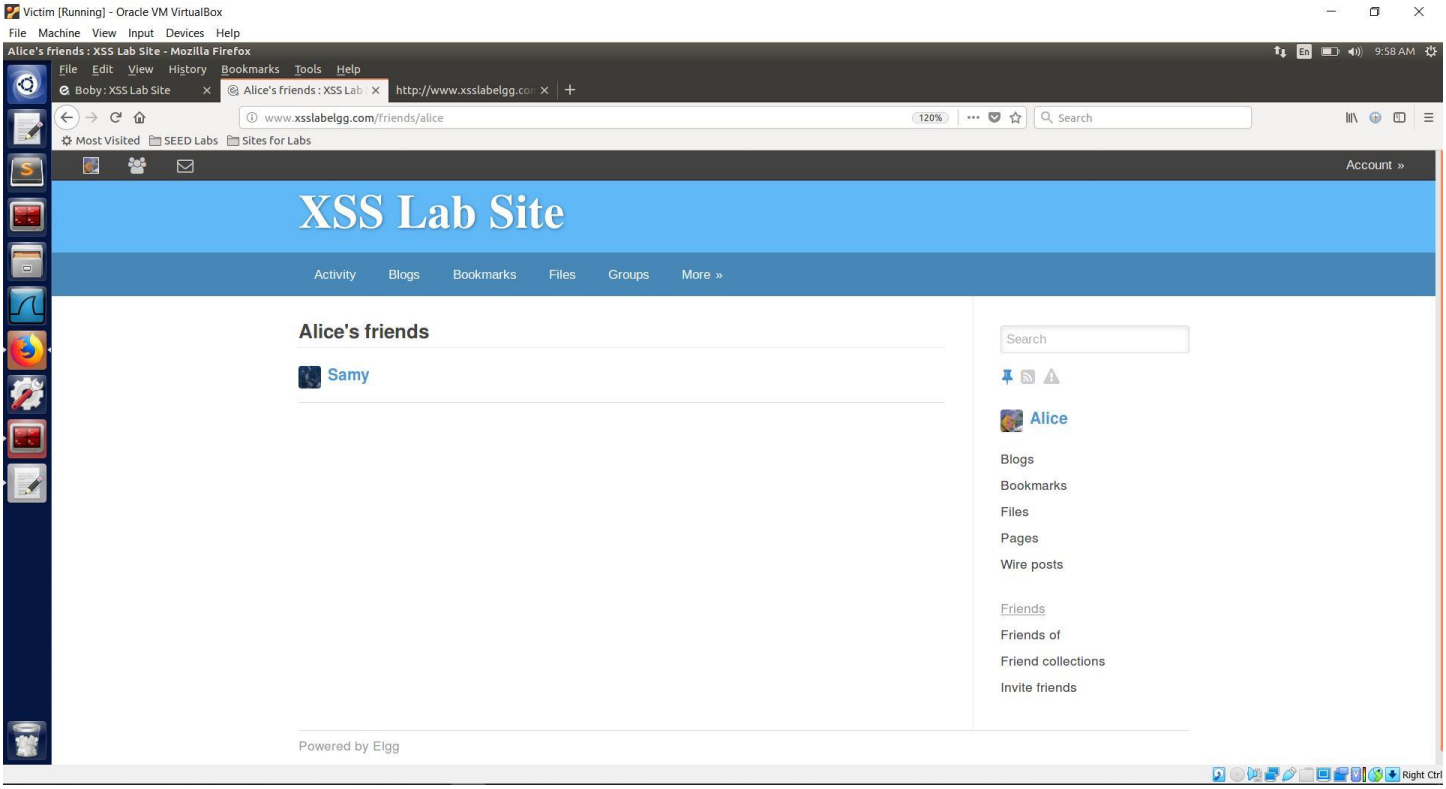
We go to the members page and then to Samy's profile.



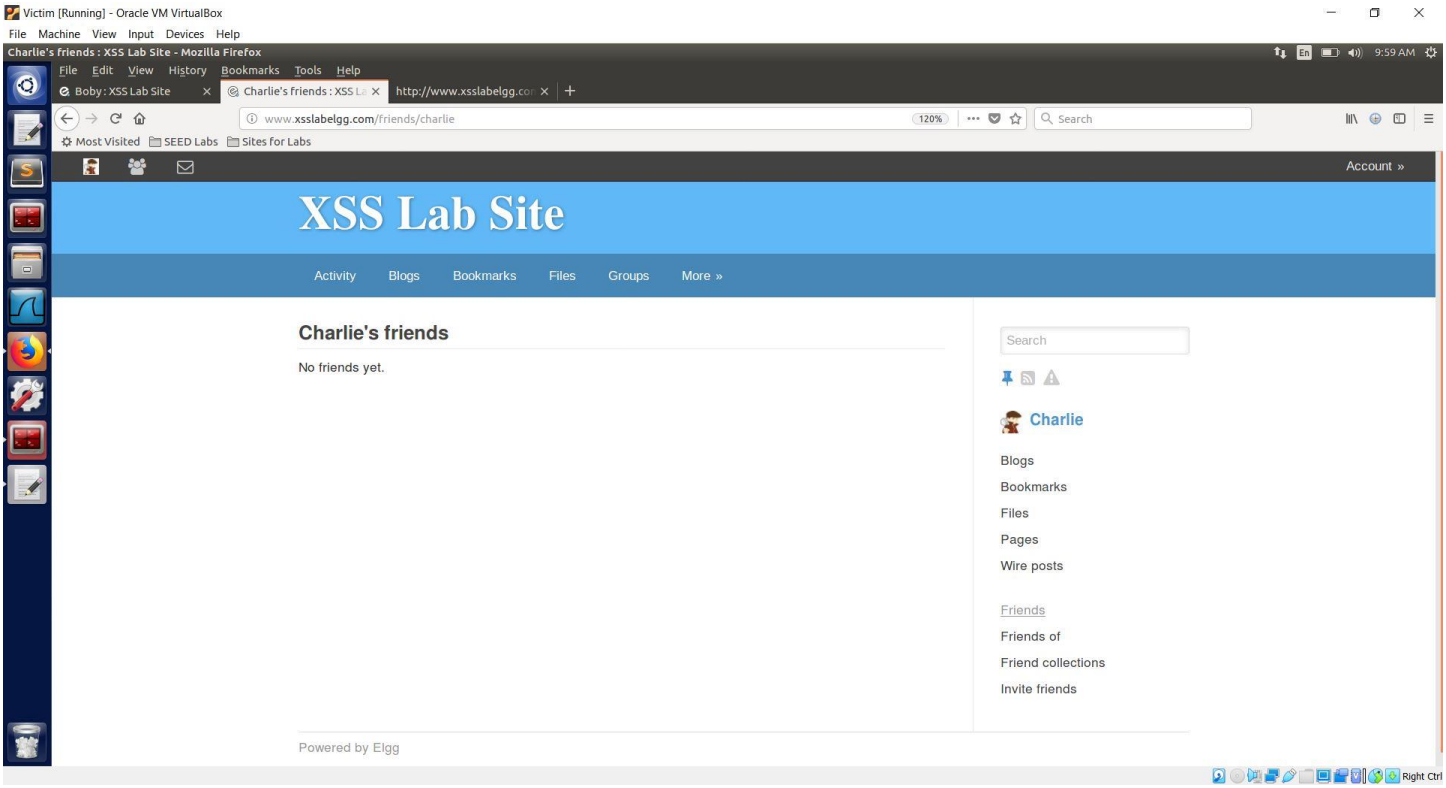


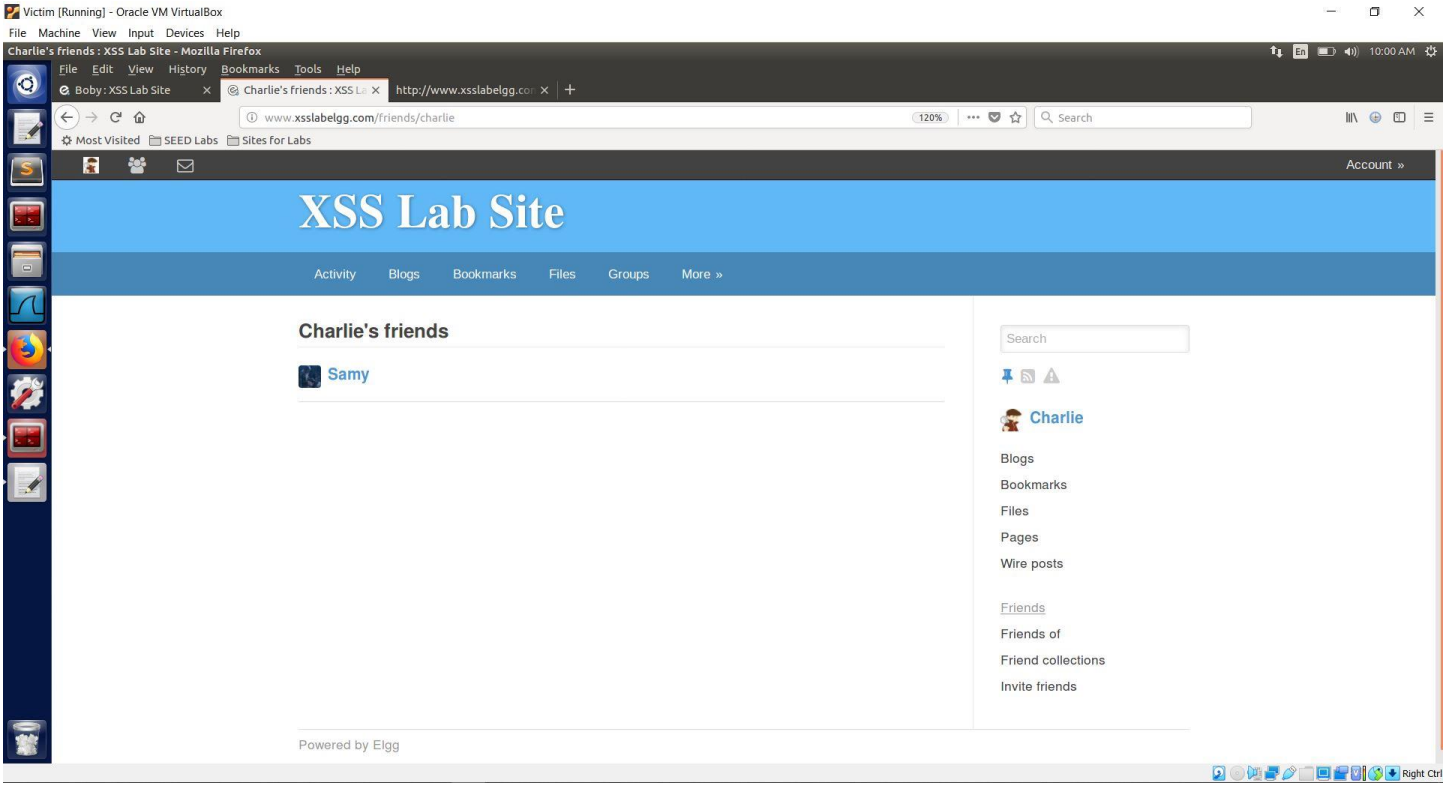
If our XSS attack worked, then Samy should be added as Alice's friend. We will go to Alice's friends page.

From the below screenshot we observe that Samy is added as Alice's friend which was earlier empty. Thus our XSS attack scripts was successfully executed.



We can verify this again with another user as our script adds Samy as a friend to any user who visits Samy's Profile. Below screenshots show another user Charlie to which the Samy is added as friend.



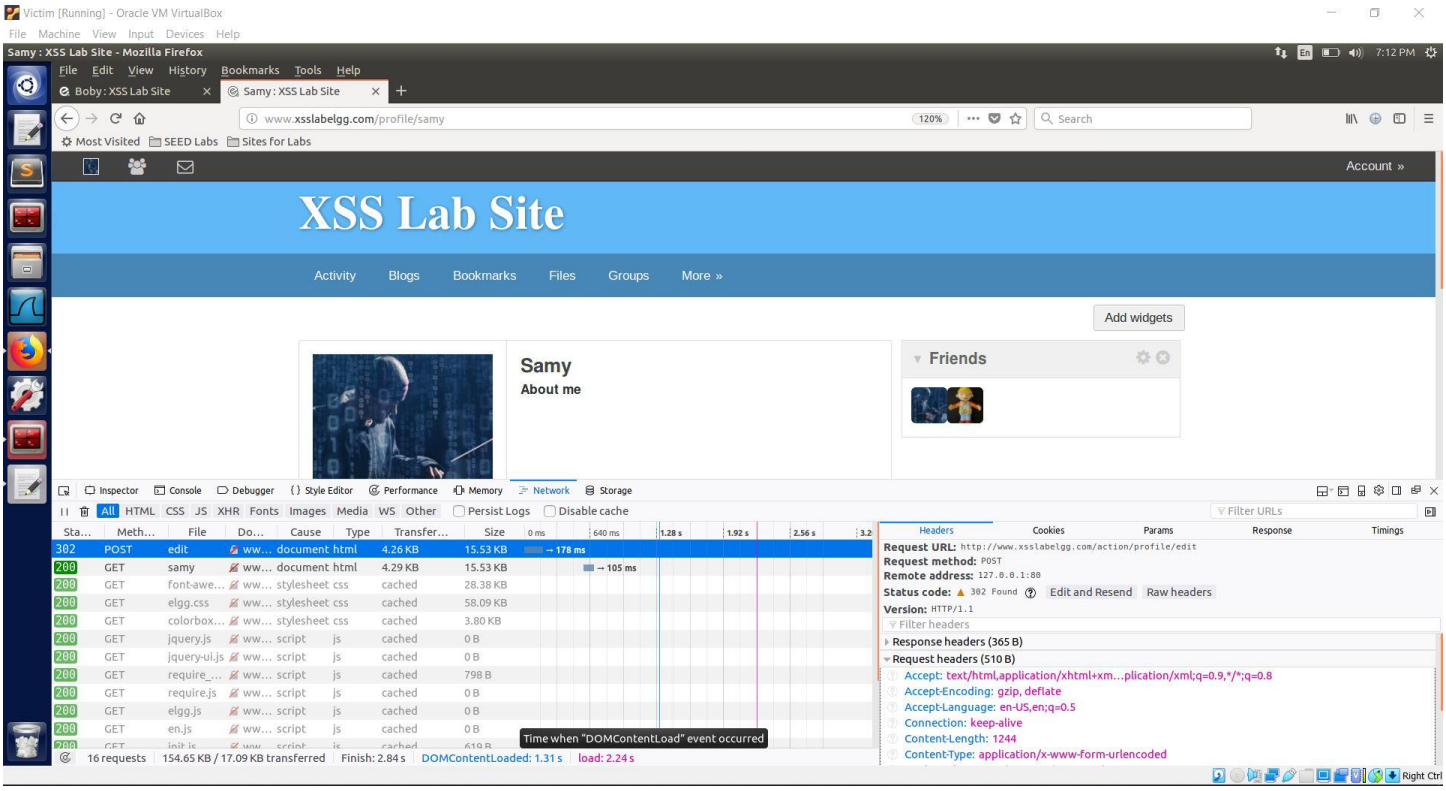


Thus, we successfully added our user Samy as the victim's friend without their knowing.

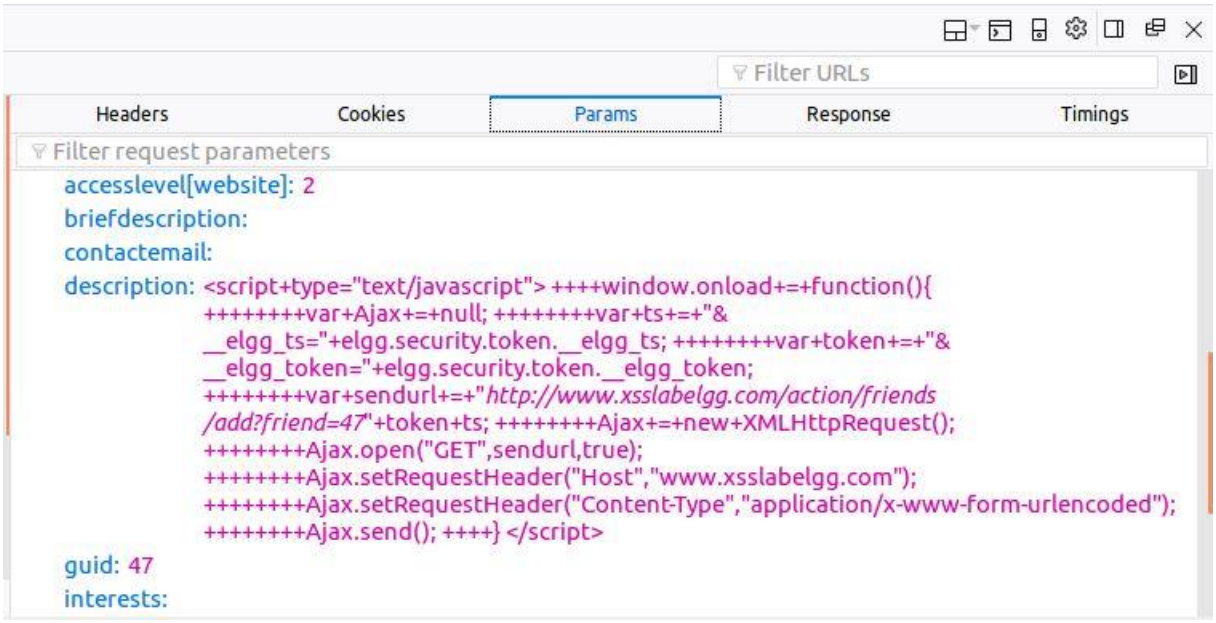
Task 5: Modifying the victim's profile

Now we will try to modify the victim's profile using a Javascript code with a POST request. First, we will need to understand the POST request sent when the profile is updated.

We will edit a profile and save to observe the request in the developer tools.

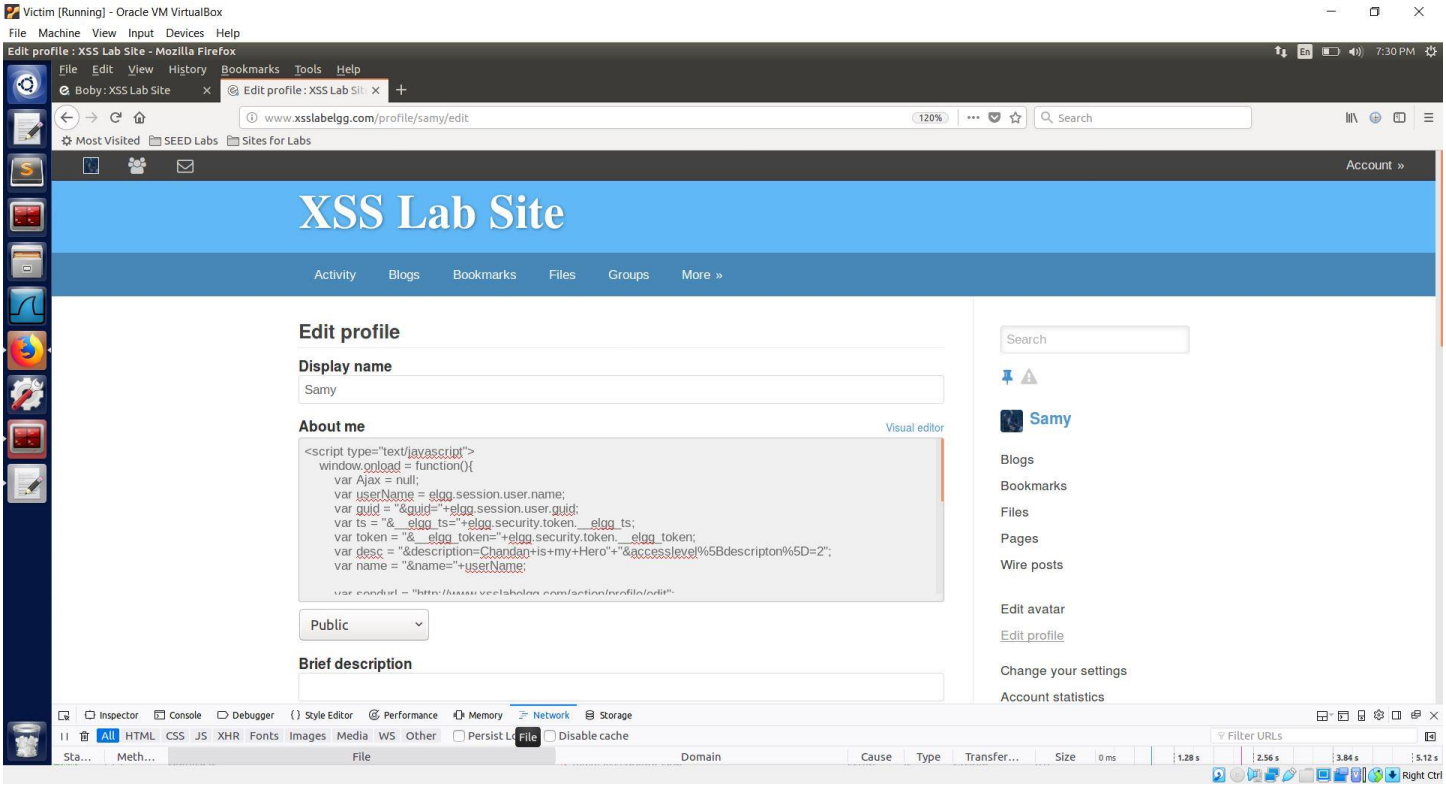


The above screenshot shows the post request sent to http://www.xsslabelgg.com/action/profile/edit when we edit the profile. The parameters sent with the POST request are shown below.

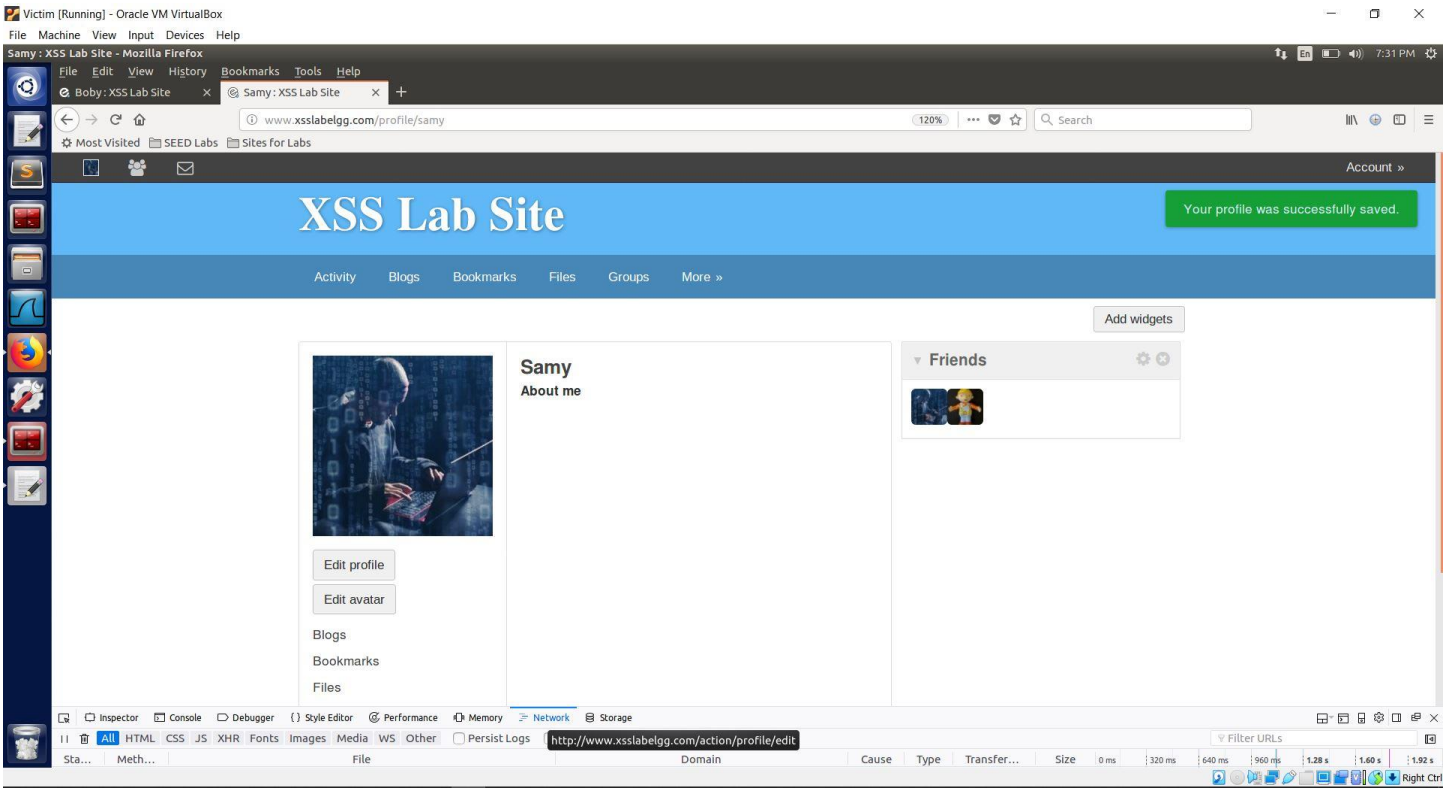


We observe that content consists of tokens and ts variables in addition to the profile fields and the access levels.

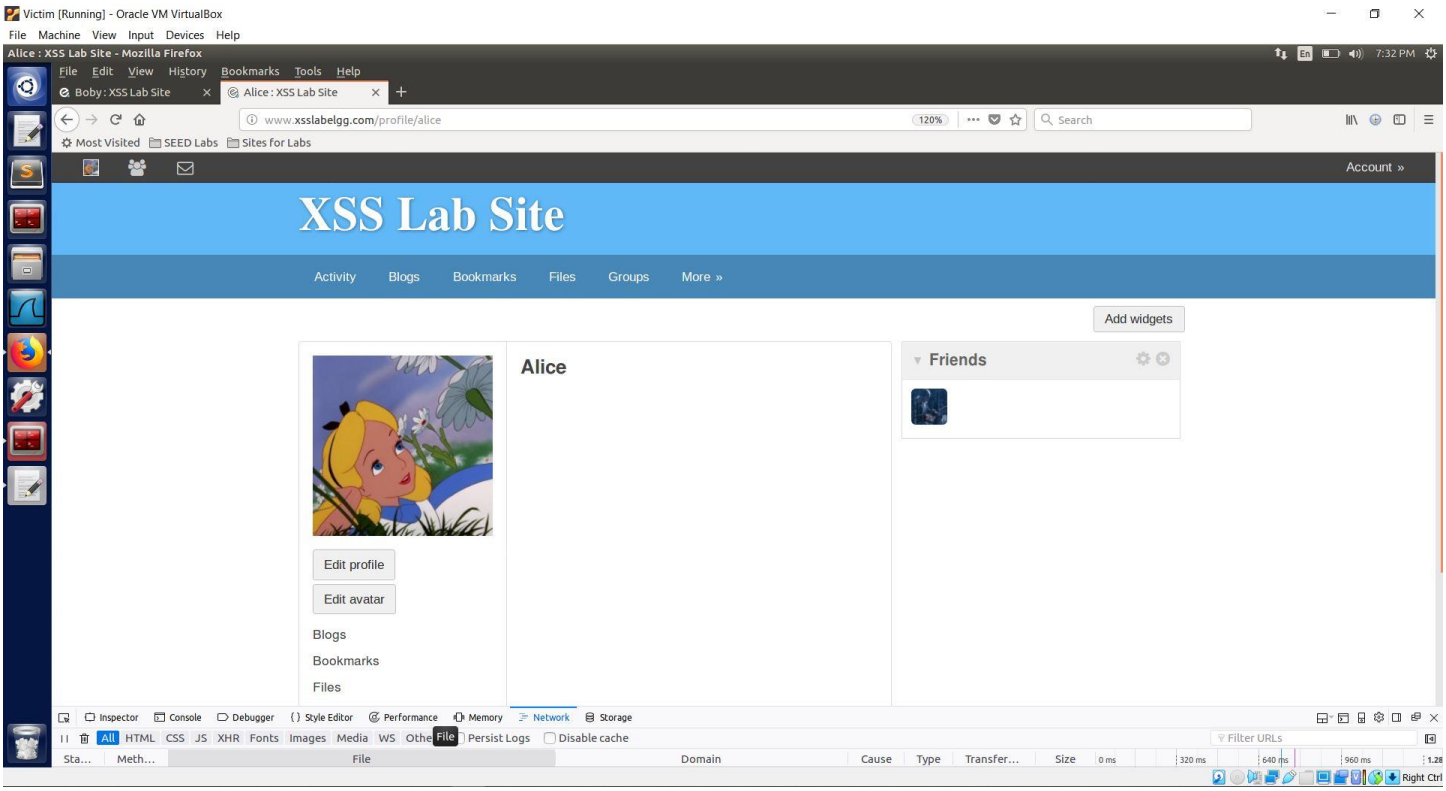
The XSS code script is shown below where we try to achieve a legitimate POST request with the parameters.



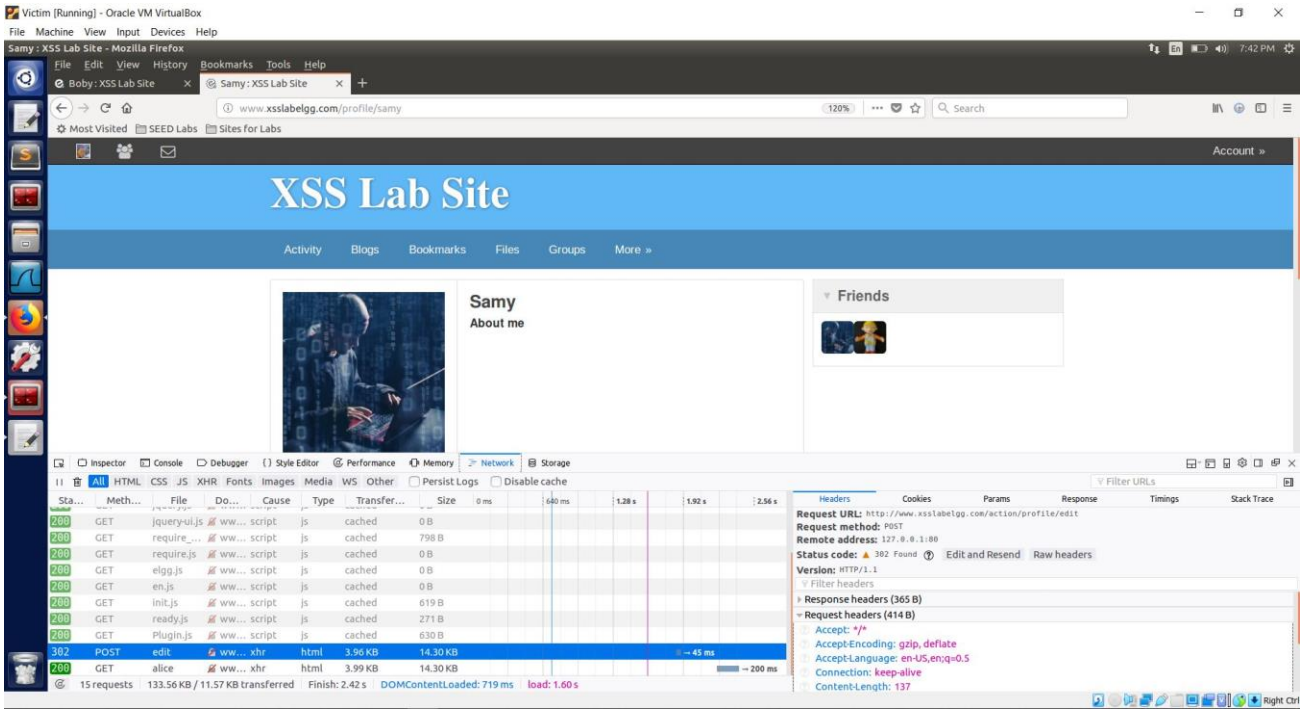
On saving the profile we don't see anything on the webpage apart from the popup of profile updated as shown below.

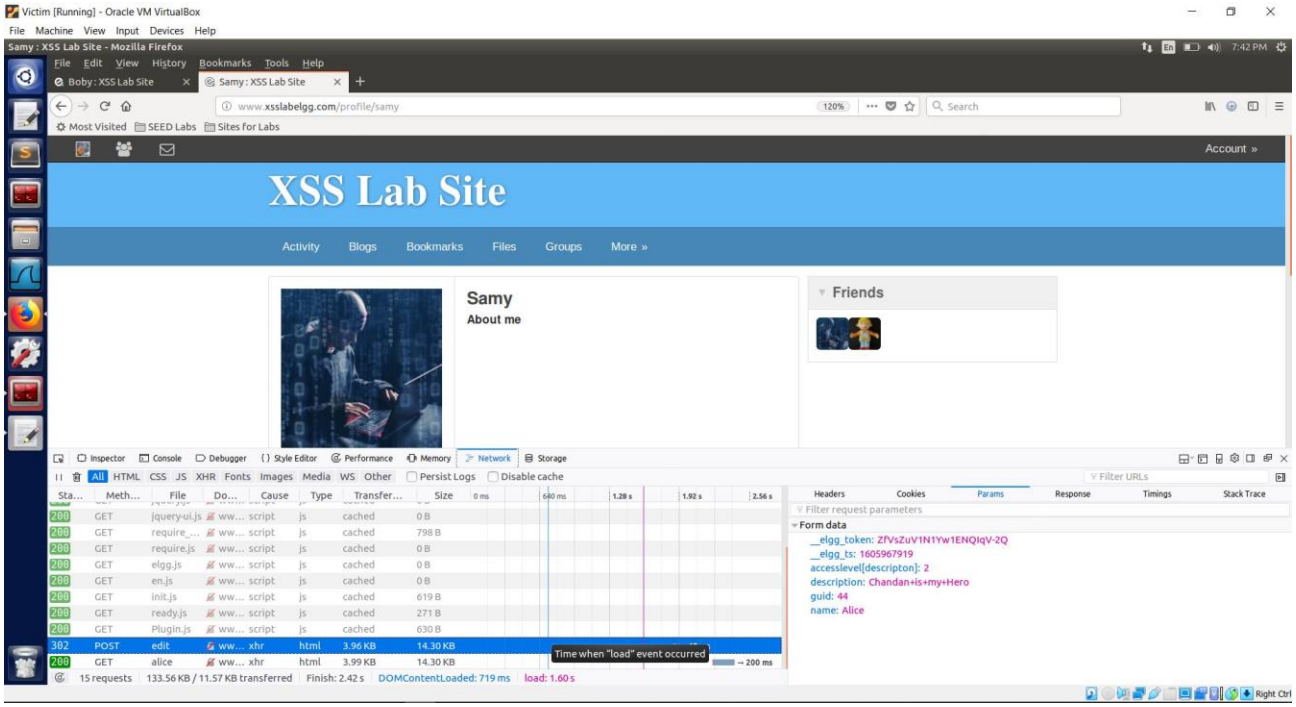


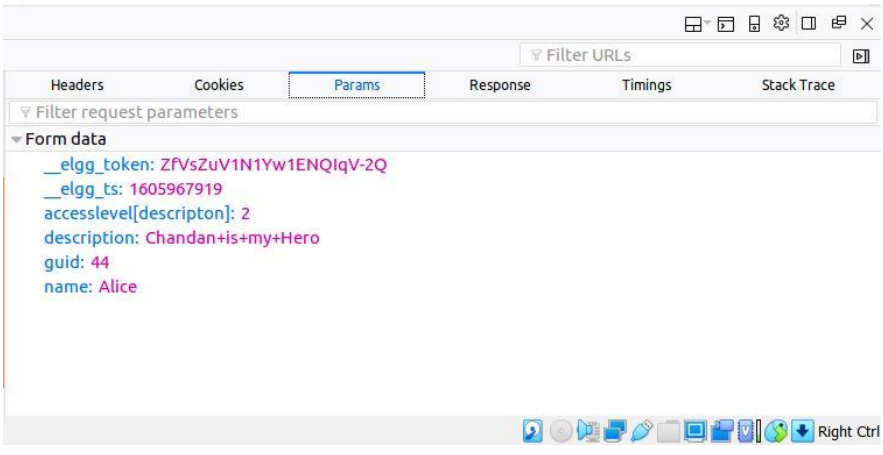
Next we login in to the website as Alice and visit Samy's profile from the members page.



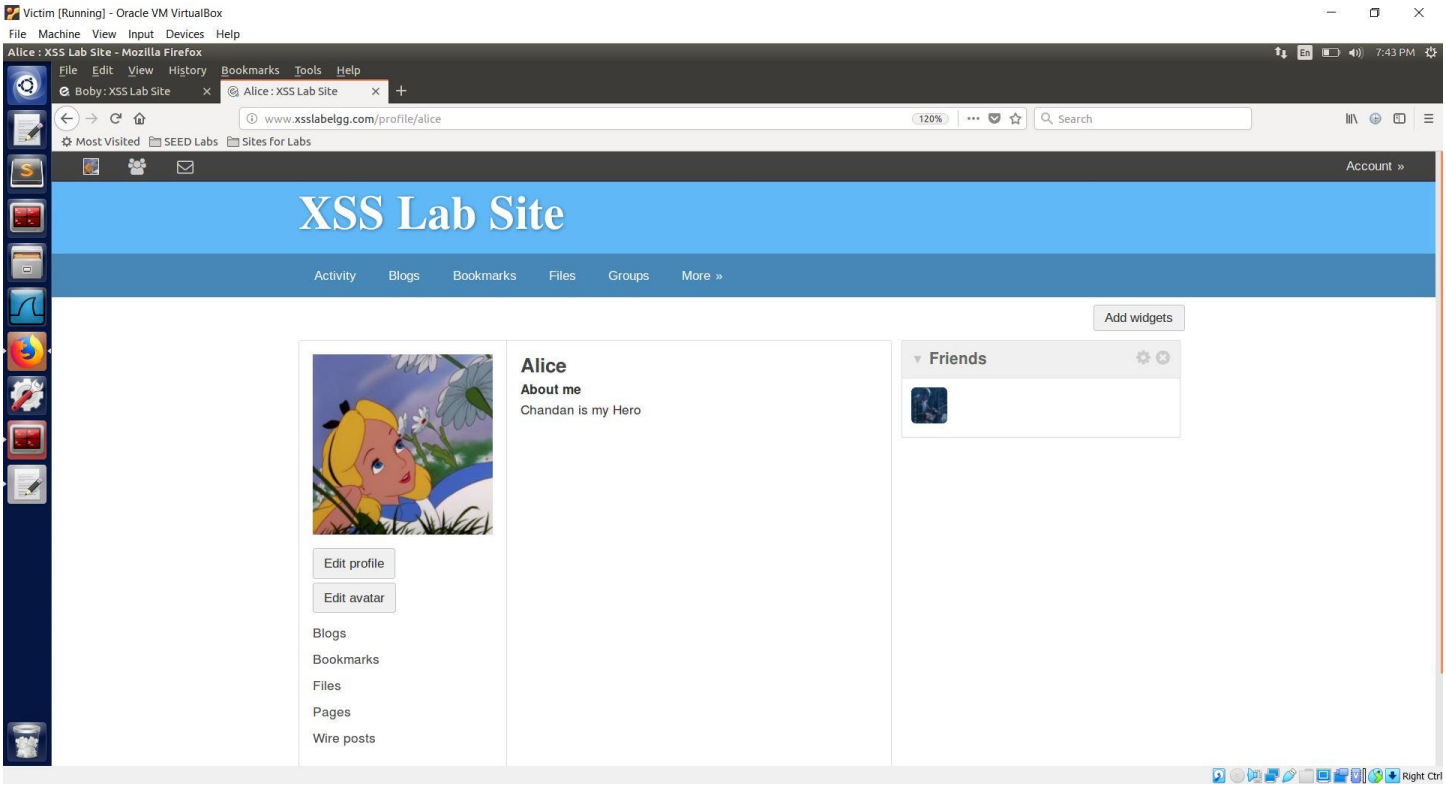
When we visit Samy's profile from Alice we see that a POST request is sent as observed in the dev tools. The POST request and the parameters are shown in the below screenshots.


Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Samy : XSS Lab Site - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Boby : XSS Lab Site
Samy : XSS Lab Site
www.xsslabelgg.com/profile/samy
Most Visited SEED Labs Sites for Labs
Account »
XSS Lab Site
Activity Blogs Bookmarks Files Groups More »
Samy
About me
Friends
Inspector Console Debugger Style Editor Performance Memory Network Storage
All HTML CSS JS XHR Fonts Images Media WS Other Persist Logs Disable cache
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found Edit and Resend Raw headers
Version: HTTP/1.1
Response headers (365 B)
Request headers (414 B)
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 137
15 requests 133.56 KB / 11.57 KB transferred Finish: 2.42 s DOMContentLoaded: 719 ms load: 1.60 s


Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Samy : XSS Lab Site - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Boby : XSS Lab Site
Samy : XSS Lab Site
www.xsslabelgg.com/profile/samy
Most Visited SEED Labs Sites for Labs
Account »
XSS Lab Site
Activity Blogs Bookmarks Files Groups More »
Samy
About me
Friends
Inspector Console Debugger Style Editor Performance Memory Network Storage
All HTML CSS JS XHR Fonts Images Media WS Other Persist Logs Disable cache
Filter request parameters
Form data
__elgg_token: ZfVsZuV1N1Yw1ENQIqV-2Q
__elgg_ts: 1605967919
accesslevel[descripton]: 2
description: Chandan+is+my+Hero
guid: 44
name: Alice
Time when "load" event occurred
15 requests 133.56 KB / 11.57 KB transferred Finish: 2.42 s DOMContentLoaded: 719 ms load: 1.60 s


Filter URLs
Headers
Cookies
Params
Response
Timings
Stack Trace
Filter request parameters
Form data
__elgg_token: ZfVsZuV1N1Yw1ENQIqV-2Q
__elgg_ts: 1605967919
accesslevel[descripton]: 2
description: Chandan+is+my+Hero
guid: 44
name: Alice
Right Ctrl

Now if our XSS attack was successful, then Alice's profile should be updated with the message we wanted "Chandan is my Hero".

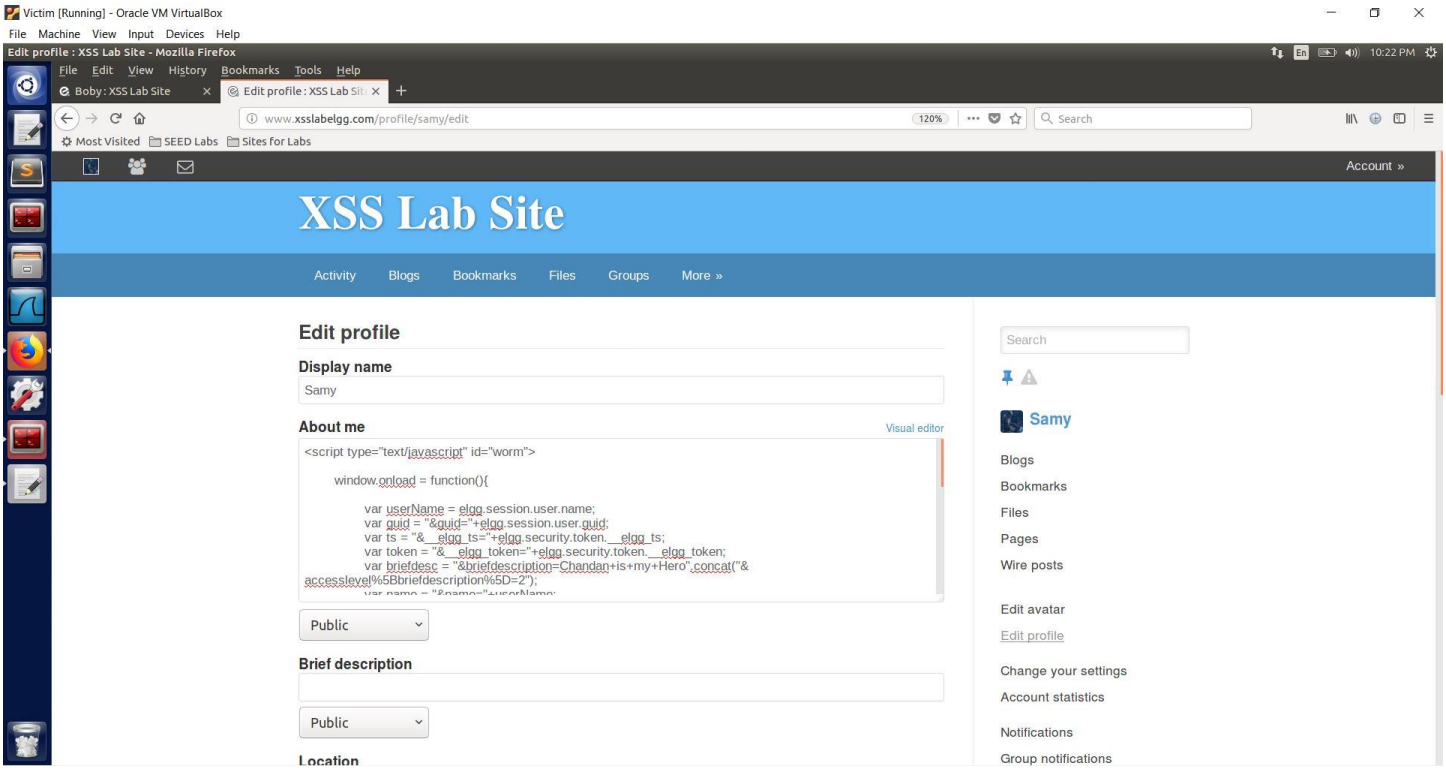


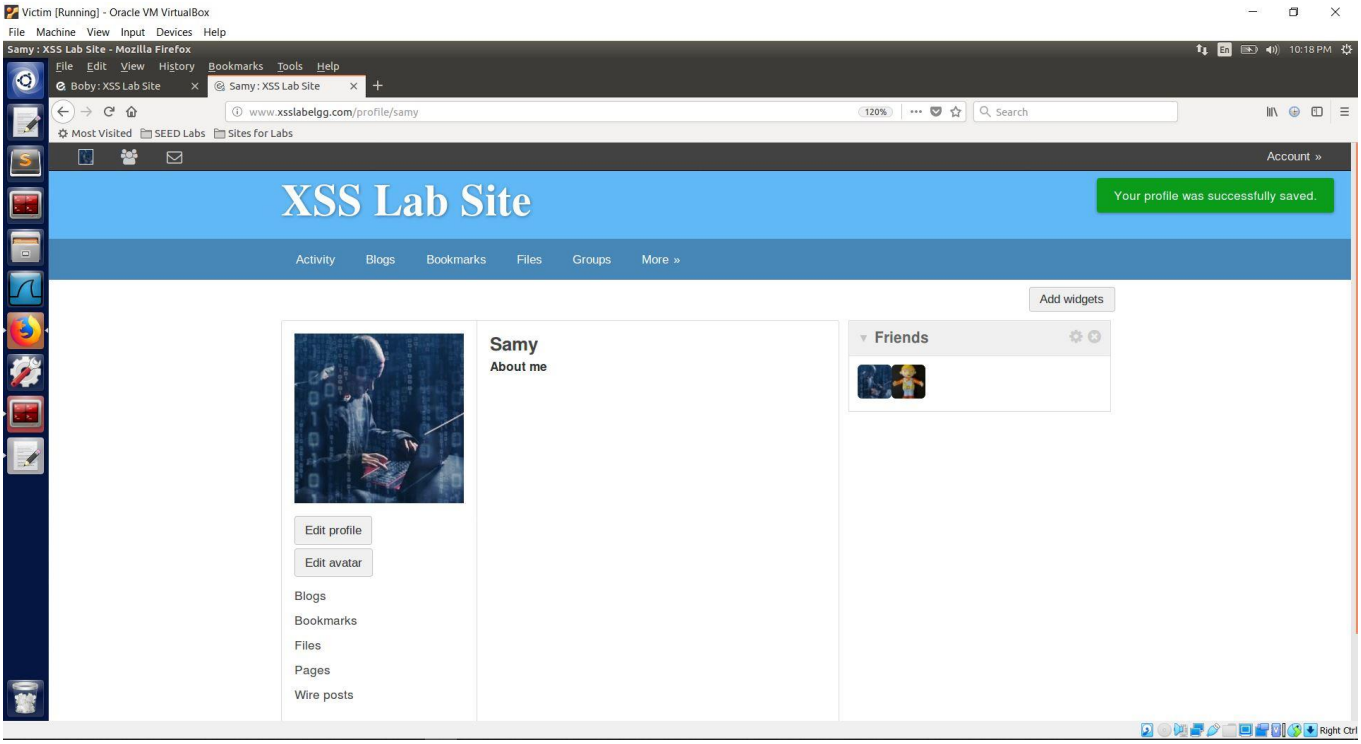
From the above screenshot it is evident that our XSS attack was successful and we were able to update the Victim's profile through the XSS attack.

Task 6: Writing a self-propagating XSS worm

Now we will try to achieve a self-propagating worm which will update the profile with our message and the script code responsible for it onto every victim's profile so that other users visiting the victim's profile will also be affected. The script code is shown in the below screenshot.

We write the above script code into the about me field of Samy's account and save the profile with new edit as shown below.

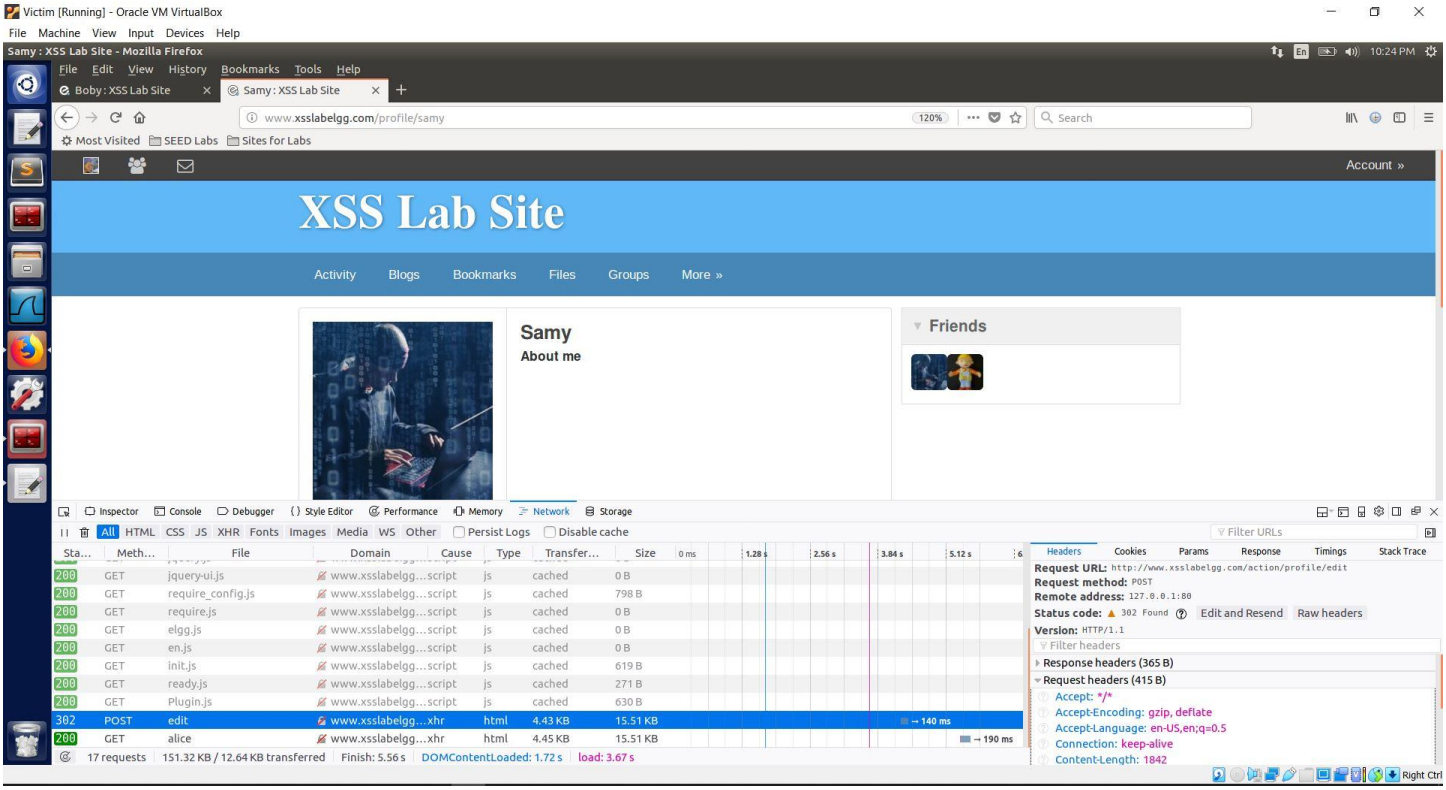


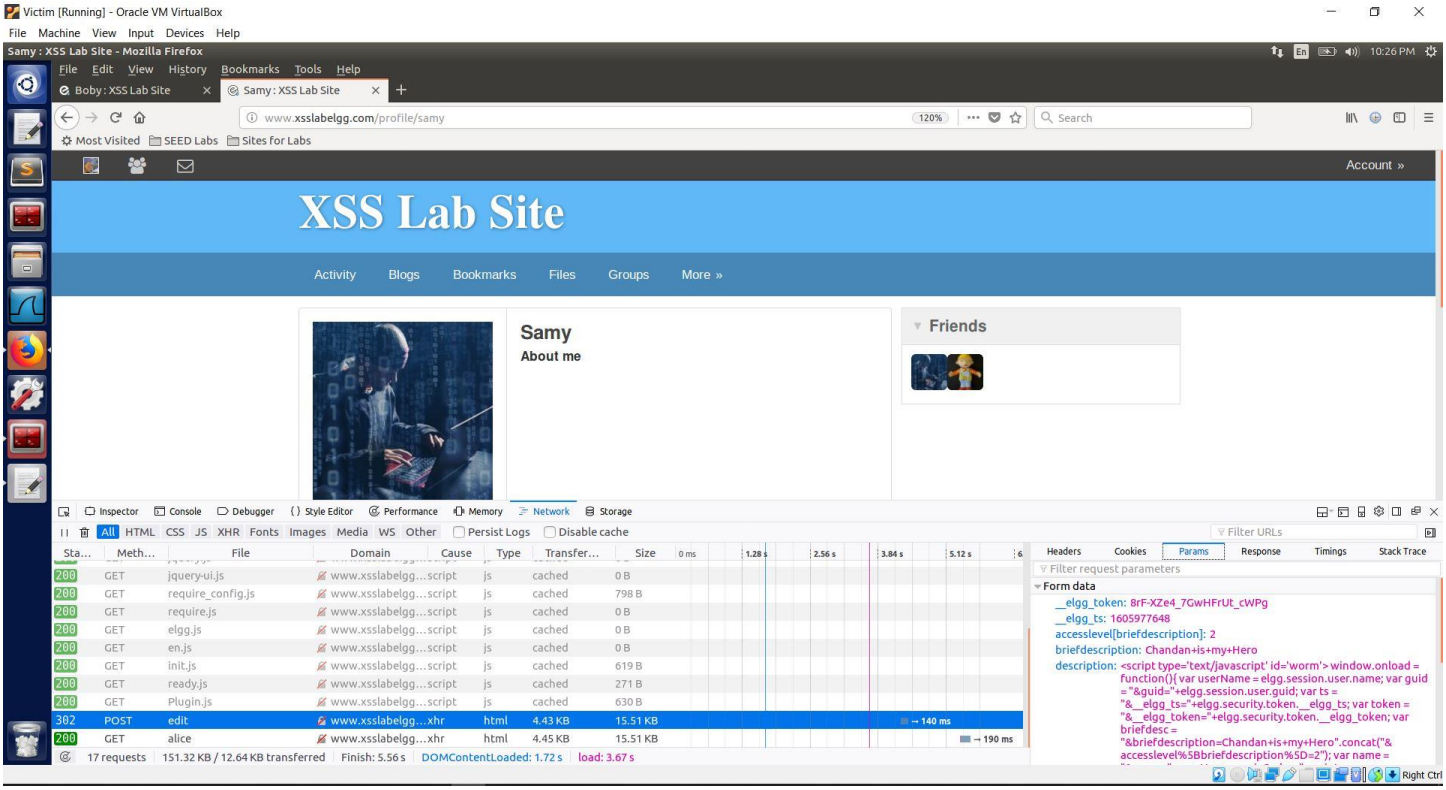


We observe a popup indicating the profile was updated.

Now we will login as Alice and visit Samy's profile. We will keep the dev tools open to observe any requests made.

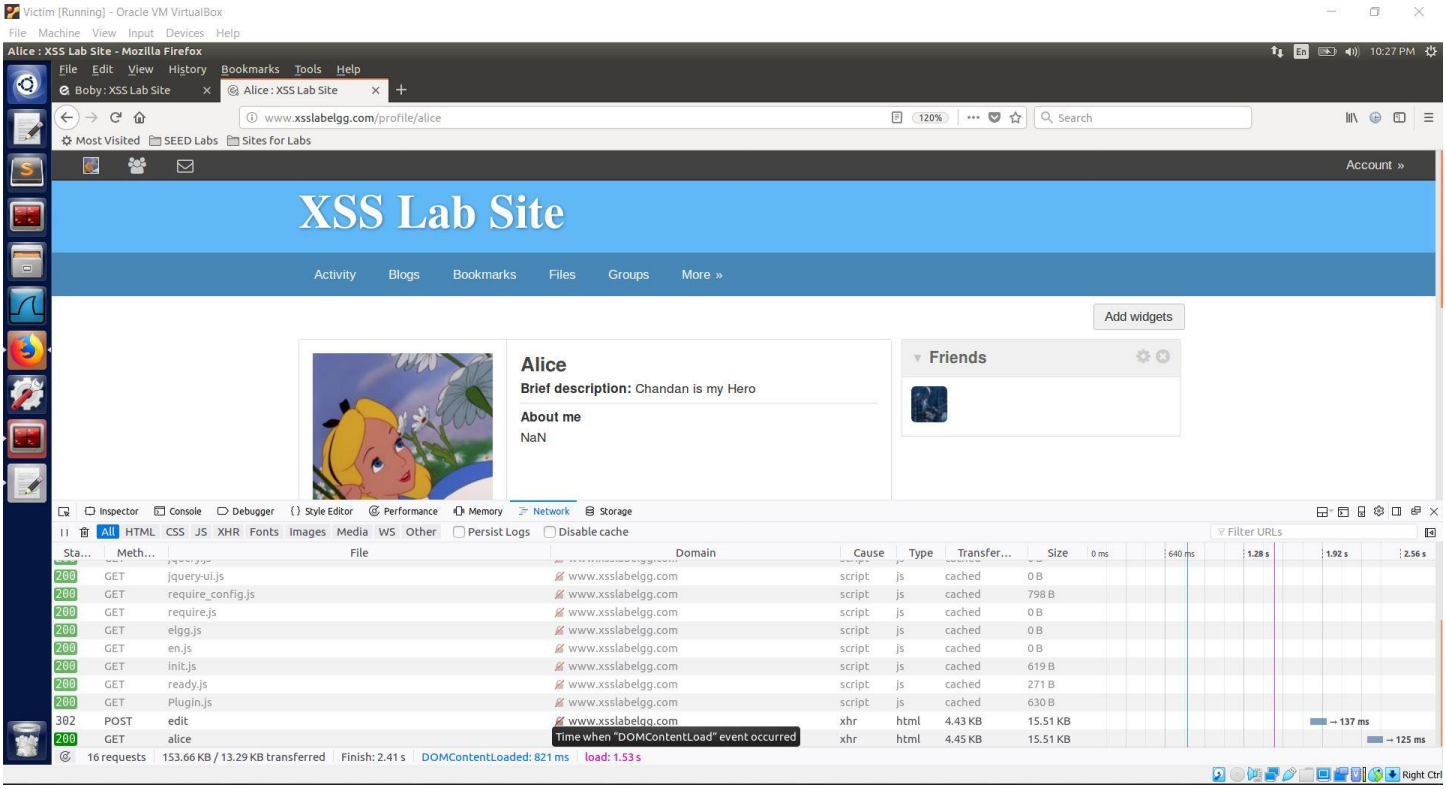
From the below screenshot observe that a POST request is made. The parameters and requesr is as follows.

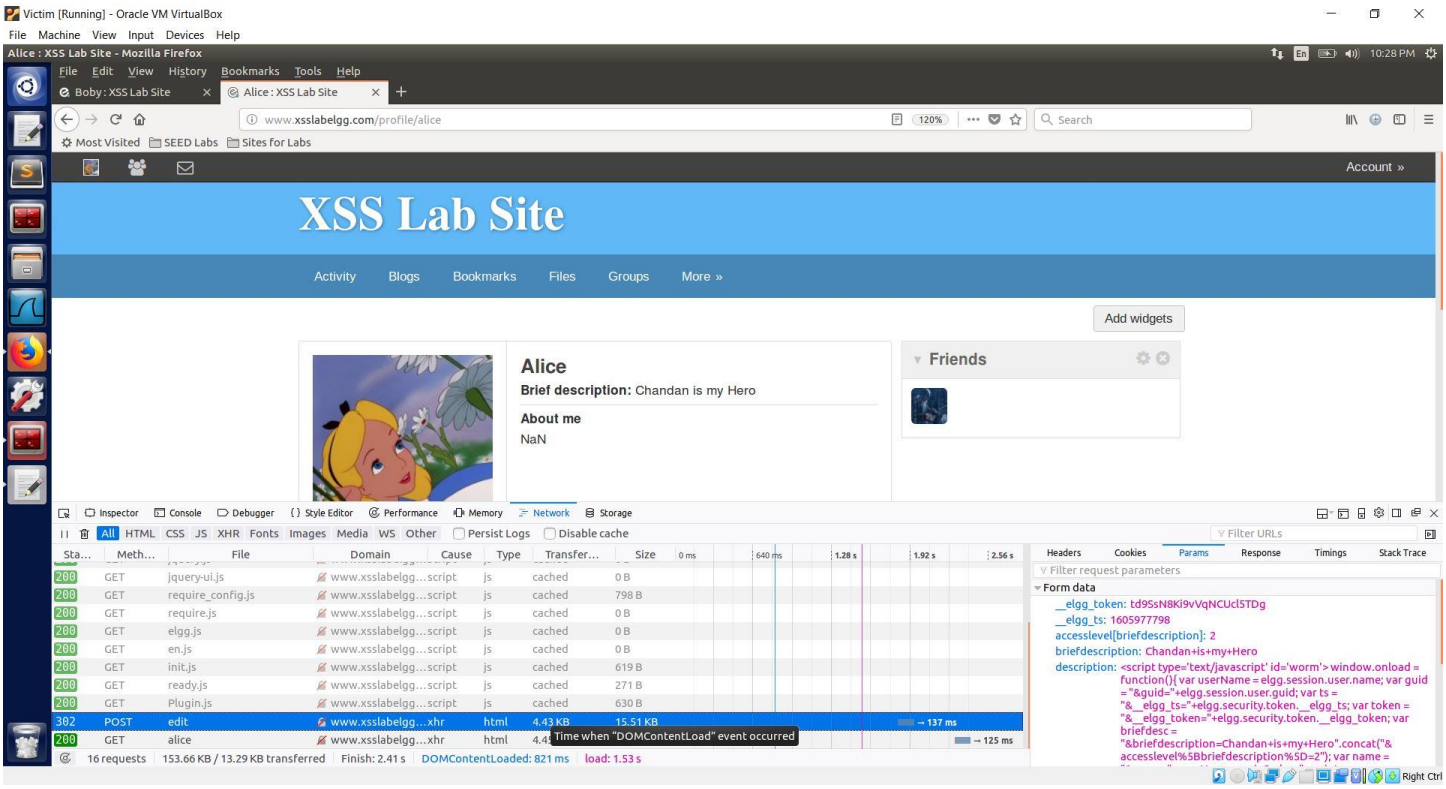




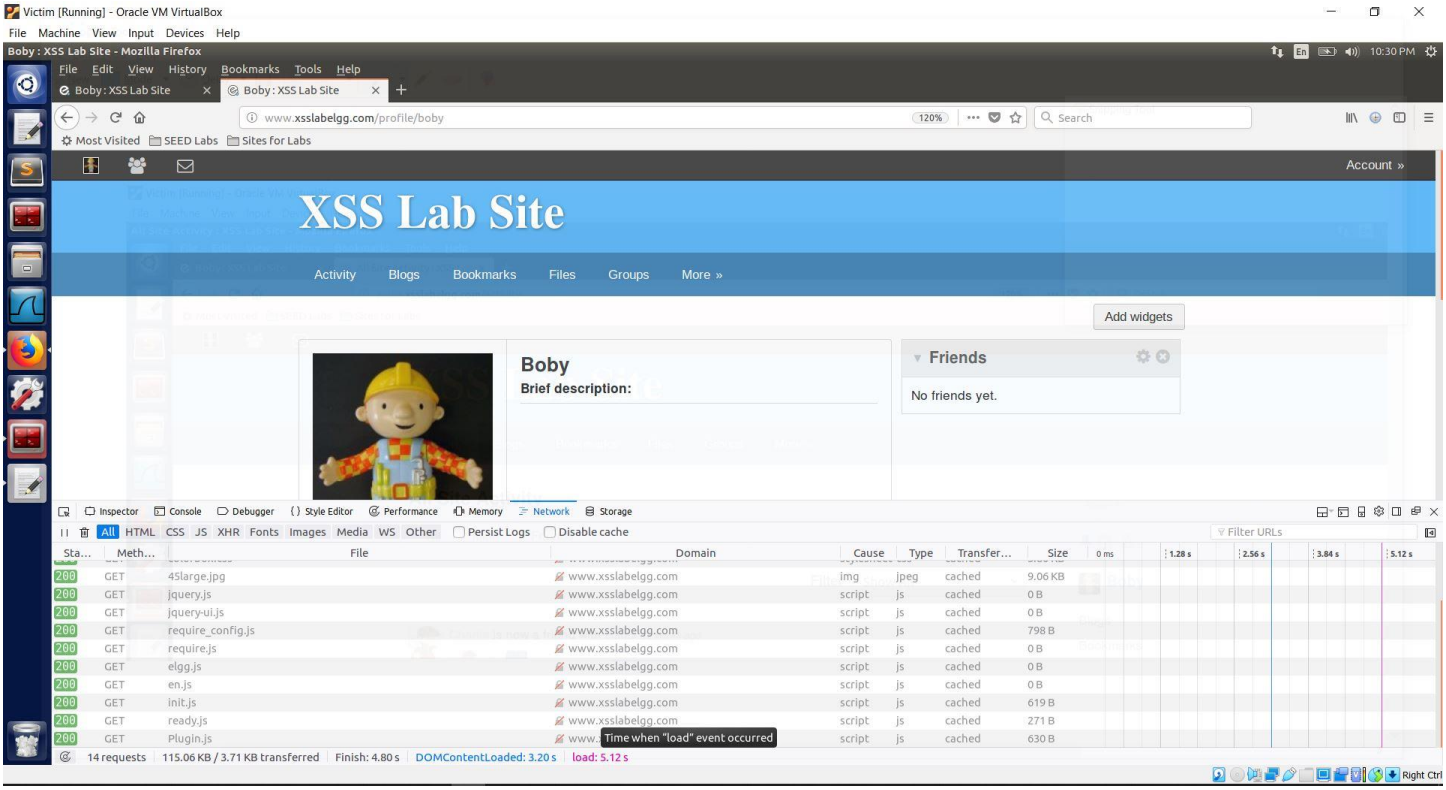
We observe that a POST request is made to /action/profile/edit. We see Alice's profile to see if it is updated.

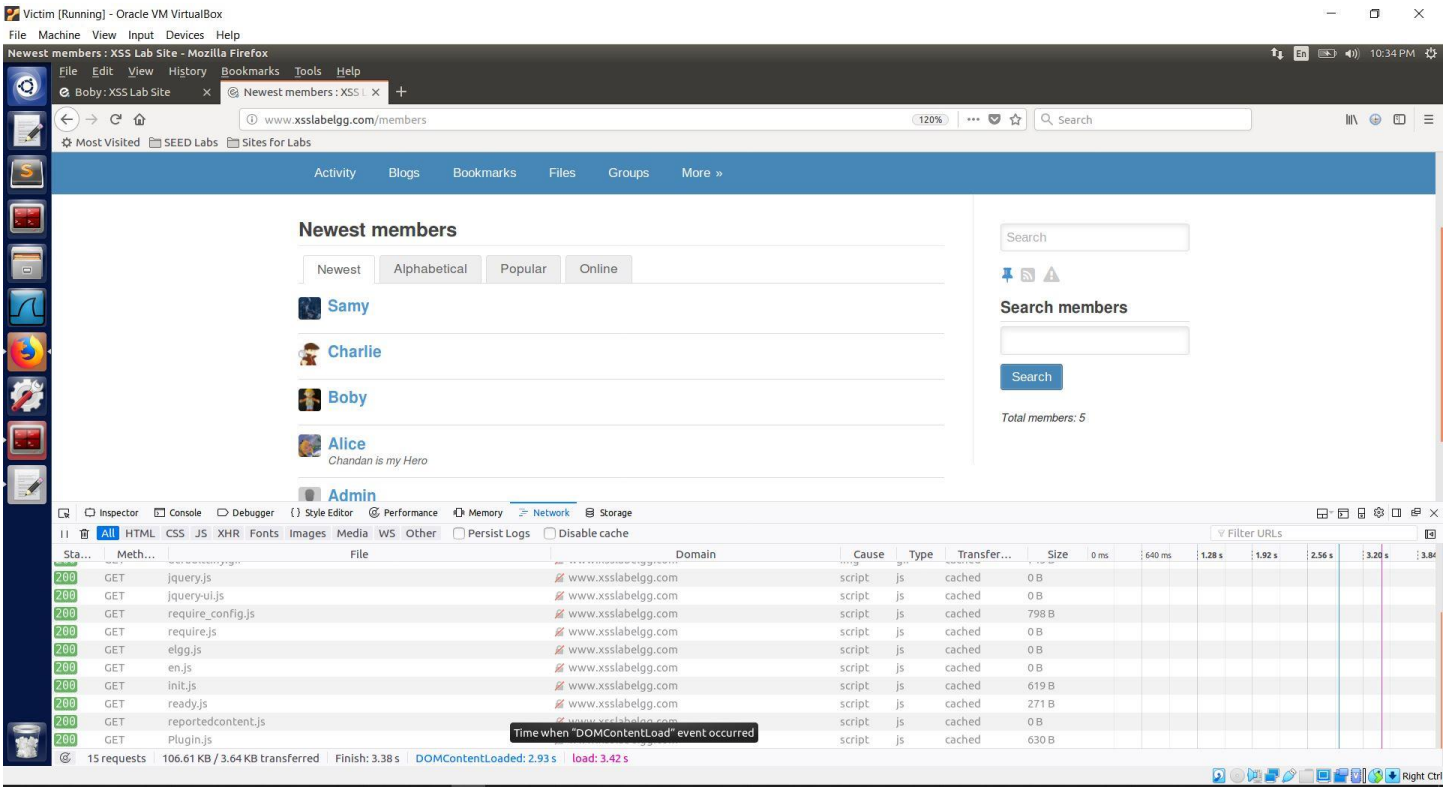
From the below screenshot we observe that the profile is updated and "Chandan is my Hero" is added as description in Alice's account. We can also observe the POST request made as shown below. Thus XSS attack was successful.

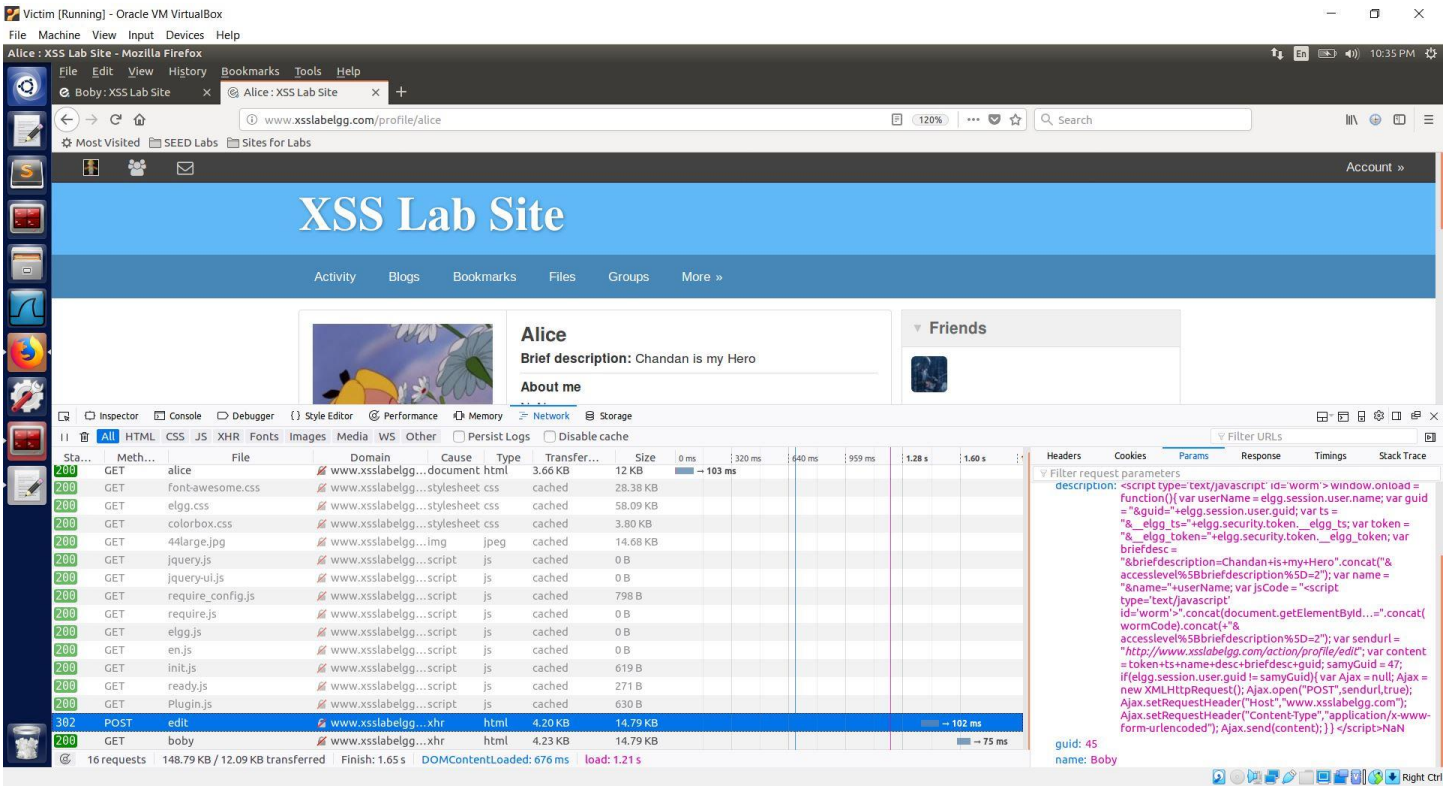


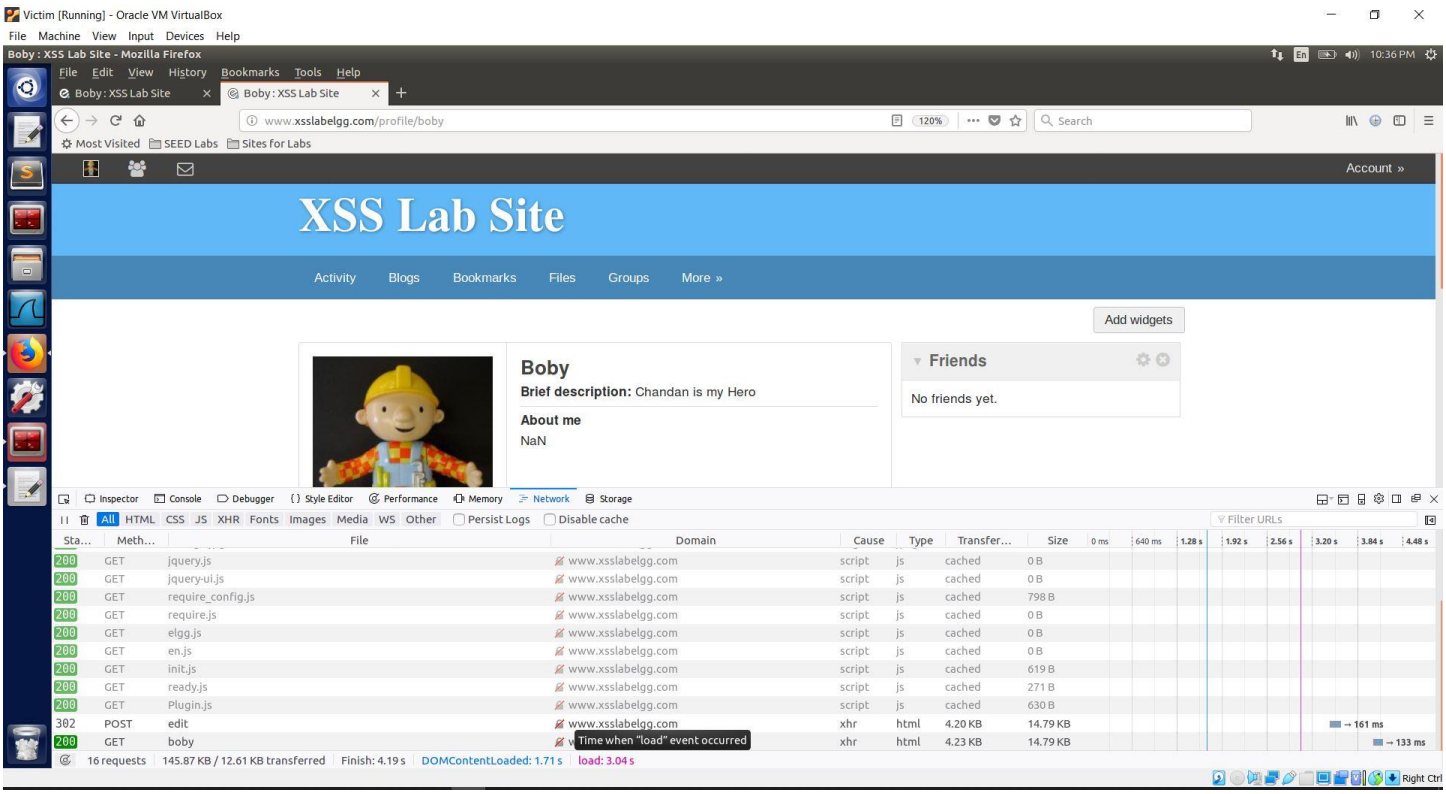


Now we login as Boby and visit Alice's profile to see if it is self-propagating. When we visit Alice's profile we observe that a POST request is made as seen in the dev tools. Thus indicating that our xss script is self-propagating. The below screenshots show the following.


Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Boby : XSS Lab Site - Mozilla Firefox
10:30 PM
File Edit View History Bookmarks Tools Help
Boby : XSS Lab Site
Boby : XSS Lab Site
www.xsslabelgg.com/profile/boby
120%
Search
Most Visited SEED Labs Sites for Labs
Account »
XSS Lab Site
Activity Blogs Bookmarks Files Groups More »
Add widgets
Boby
Brief description:
Friends
No friends yet.
Inspector Console Debugger Style Editor Performance Memory Network Storage
All HTML CSS JS XHR Fonts Images Media WS Other Persist Logs Disable cache
Filter URLs
Sta... Meth... File Domain Cause Type Transfer... Size
200 GET 45large.jpg www.xsslabelgg.com img jpeg cached 9.06 KB
200 GET jquery.js www.xsslabelgg.com script js cached 0 B
200 GET jquery-ui.js www.xsslabelgg.com script js cached 0 B
200 GET require_config.js www.xsslabelgg.com script js cached 798 B
200 GET require.js www.xsslabelgg.com script js cached 0 B
200 GET elgg.js www.xsslabelgg.com script js cached 0 B
200 GET en.js www.xsslabelgg.com script js cached 0 B
200 GET init.js www.xsslabelgg.com script js cached 619 B
200 GET ready.js www.xsslabelgg.com script js cached 271 B
200 GET Plugin.js www.xsslabelgg.com script js cached 630 B
Time when "load" event occurred
14 requests 115.06 KB / 3.71 KB transferred Finish: 4.80 s DOMContentLoaded: 3.20 s load: 5.12 s
Right Ctrl


Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Newest members : XSS Lab Site - Mozilla Firefox
10:34 PM
File Edit View History Bookmarks Tools Help
Boby : XSS Lab Site
Newest members : XSS
www.xsslabelgg.com/members
120%
Search
Most Visited SEED Labs Sites for Labs
Activity Blogs Bookmarks Files Groups More »
Newest members
Newest Alphabetical Popular Online
Samy
Charlie
Boby
Alice
Chandan is my Hero
Admin
Search
Search members
Search
Total members: 5
Inspector Console Debugger Style Editor Performance Memory Network Storage
All HTML CSS JS XHR Fonts Images Media WS Other Persist Logs Disable cache
Filter URLs
Sta... Meth... File Domain Cause Type Transfer... Size
200 GET jquery.js www.xsslabelgg.com script js cached 0 B
200 GET jquery-ui.js www.xsslabelgg.com script js cached 0 B
200 GET require_config.js www.xsslabelgg.com script js cached 798 B
200 GET require.js www.xsslabelgg.com script js cached 0 B
200 GET elgg.js www.xsslabelgg.com script js cached 0 B
200 GET en.js www.xsslabelgg.com script js cached 0 B
200 GET init.js www.xsslabelgg.com script js cached 619 B
200 GET ready.js www.xsslabelgg.com script js cached 271 B
200 GET reportedcontent.js www.xsslabelgg.com script js cached 0 B
200 GET Plugin.js www.xsslabelgg.com script js cached 630 B
Time when "DOMContentLoad" event occurred
15 requests 106.61 KB / 3.64 KB transferred Finish: 3.38 s DOMContentLoaded: 2.93 s load: 3.42 s
Right Ctrl





The above screenshot shows that even Boby's profile is updated with "Chandan is my Hero" and a POST request with Boby's details were sent when Alice's profile was visited who was a victim to Samy's XSS code.

Thus we were successful in self-propagating the XSS worm from victim to victim.

Task 7: Countermeasures

Now we will explore ways to counter the XSS vulnerability.

There are two ways with respect to Elgg website

1. HTMLawed 1.8

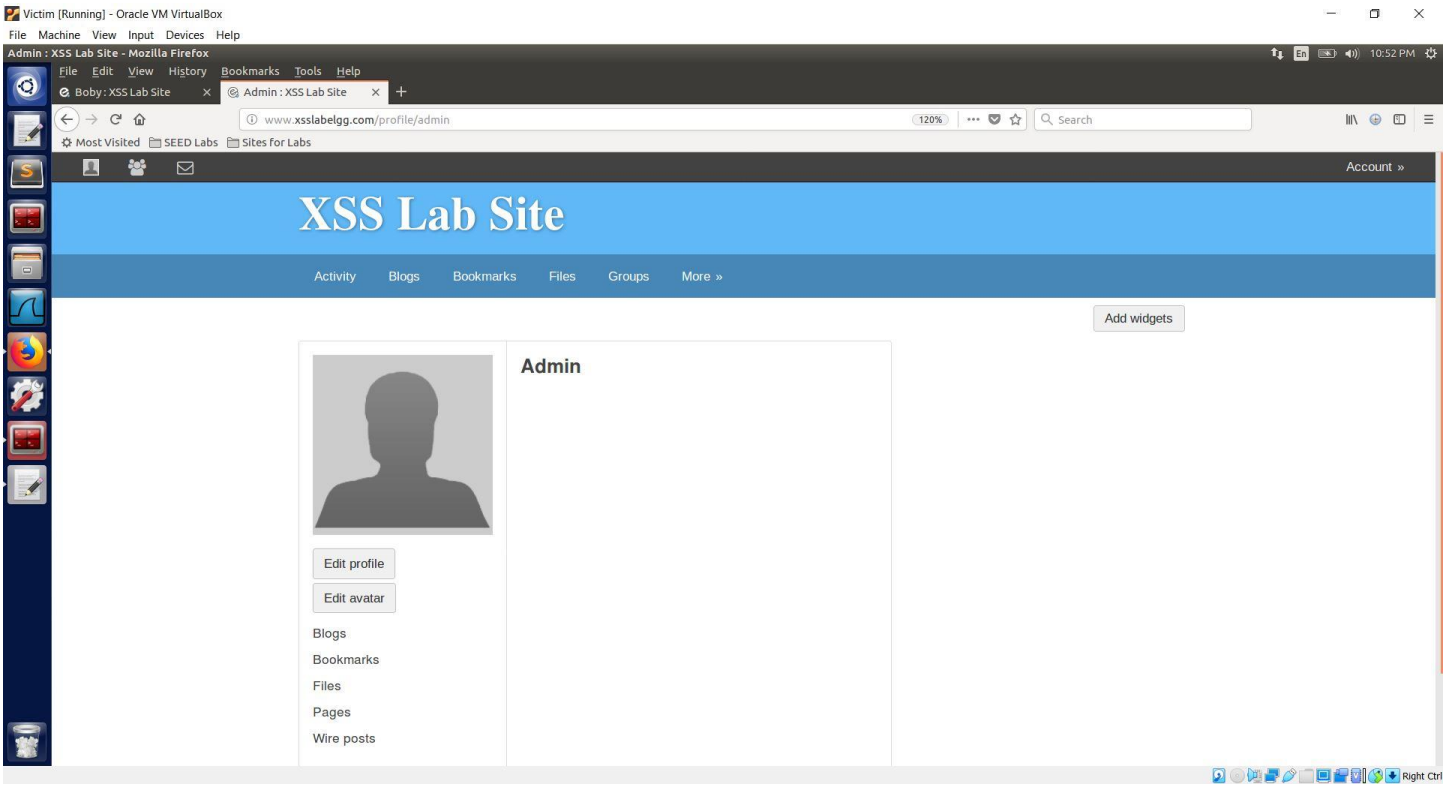
   It is a custom built security plugin on the Elgg web application which on activation, validates the user input and removes the tags from the input. This specific plugin is registered to the function filter tags in the elgg/engine/lib/input.php file.

2. Htmlspecialchars

   It is a php built in function which is used to encode the special characters in the user input, such as encod- ing "<" to , etc.

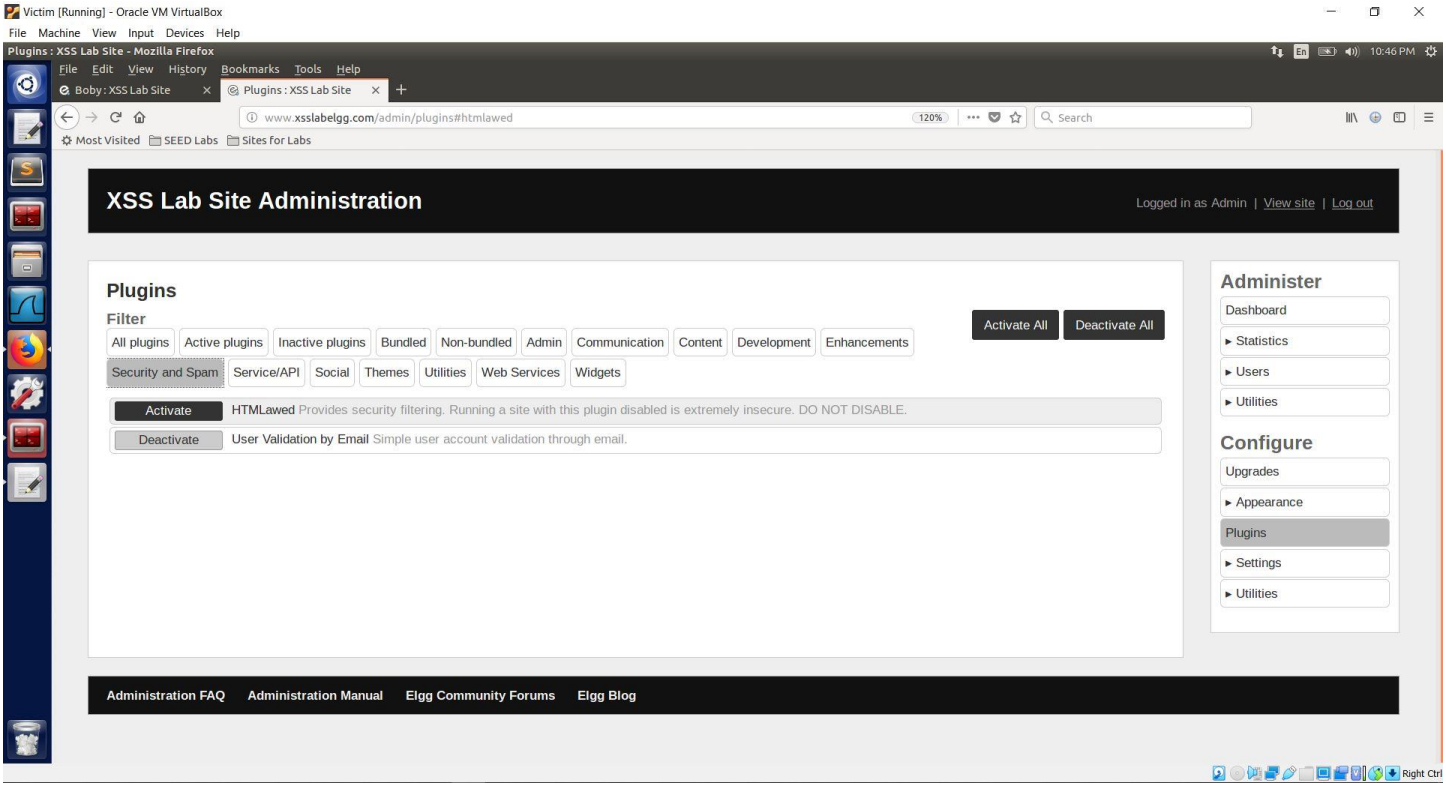
Here we will try to implement the first countermeasure.

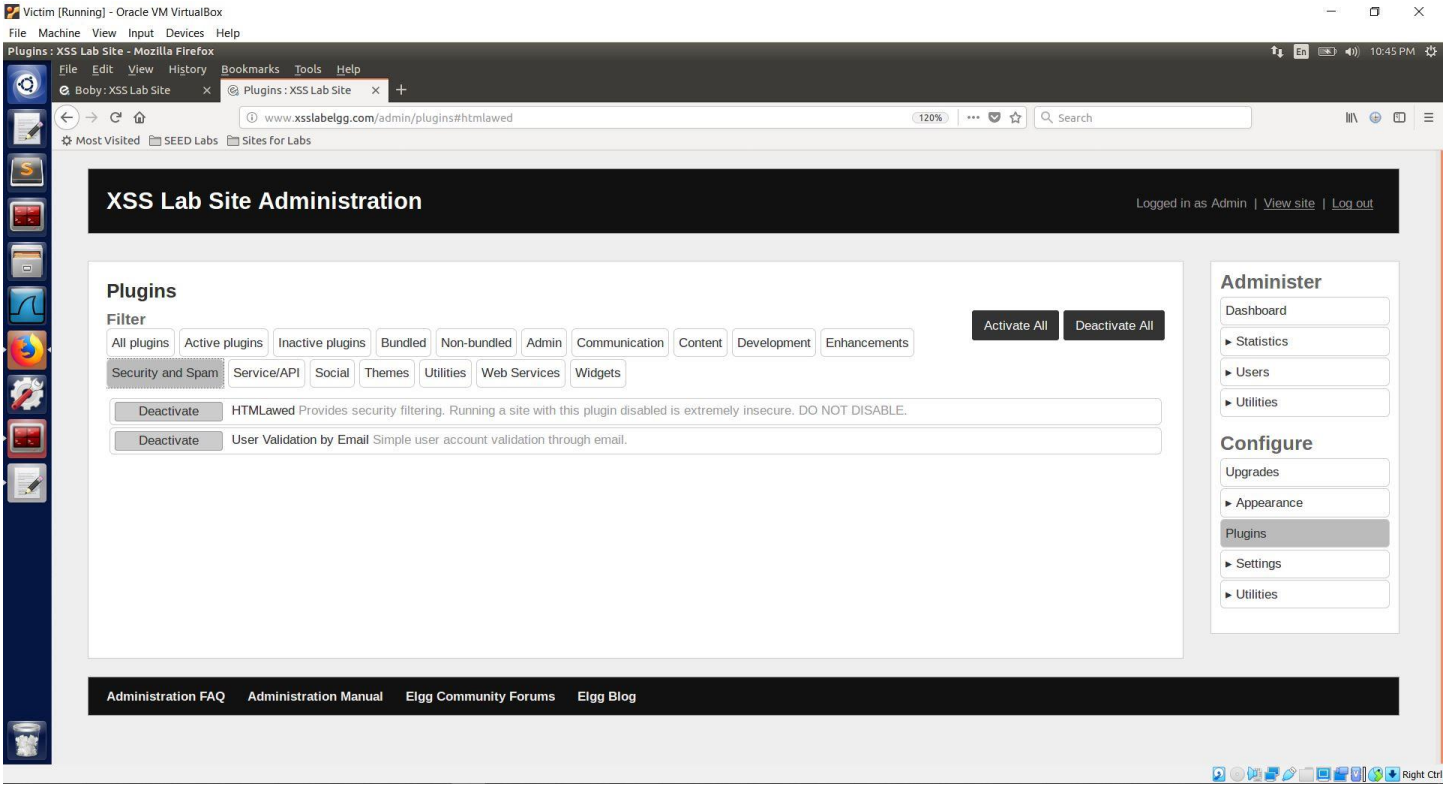
We will login to the elgg website as admin with username “admin” and password “seedelgg”.



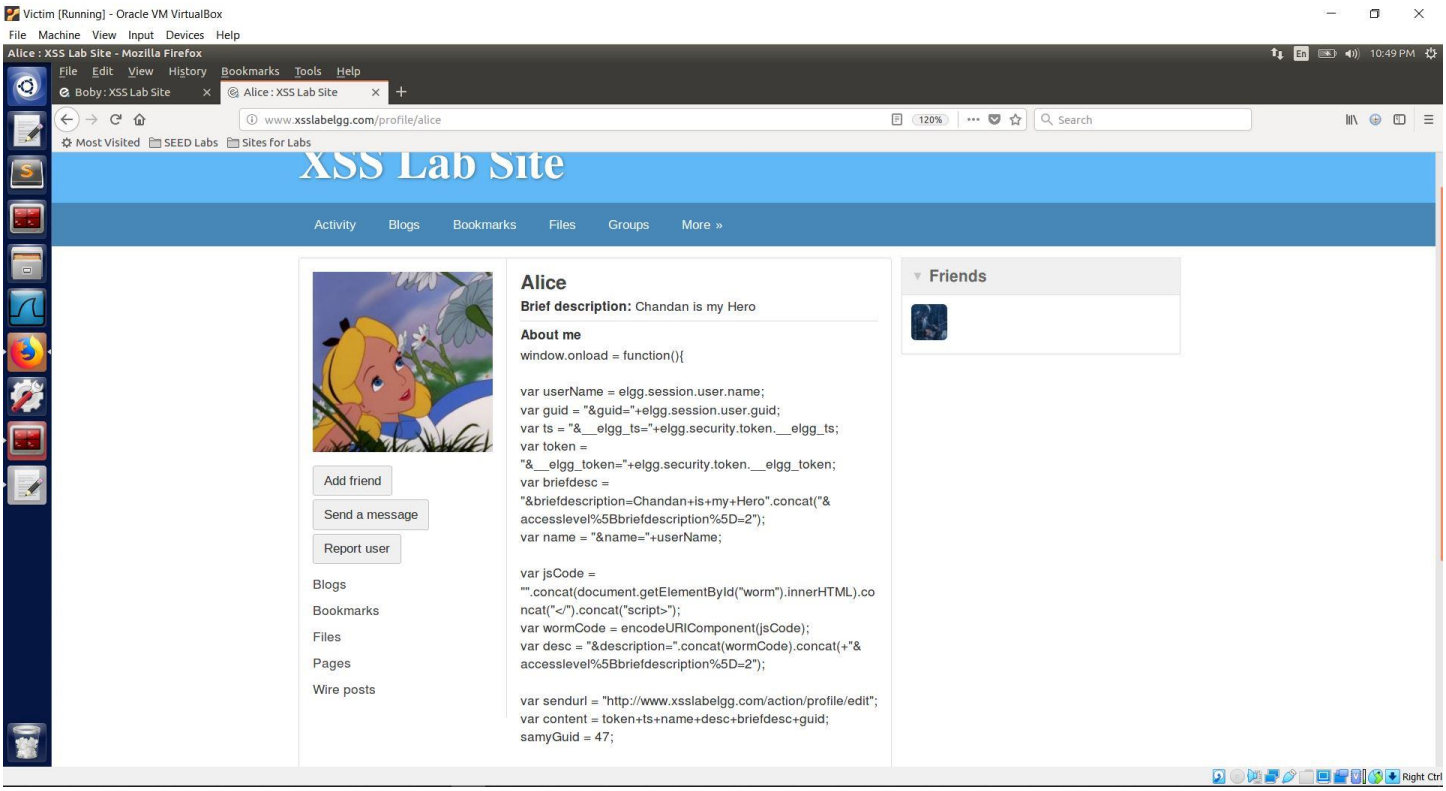
We navigate to accounts>administration>plugins and select “security and spam” as shown below.

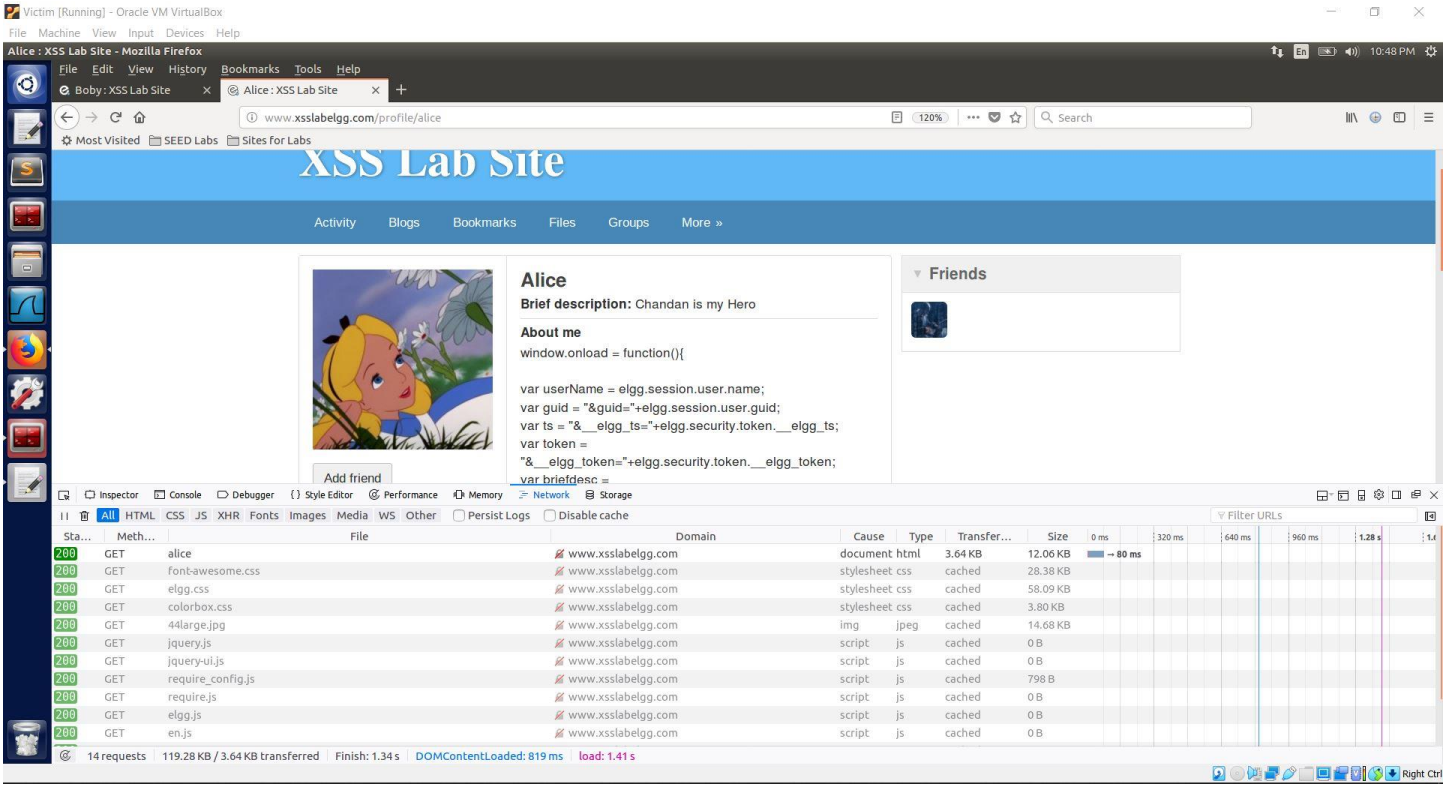
By default it is not activated. We activate the plugin as shown in the below screenshot.



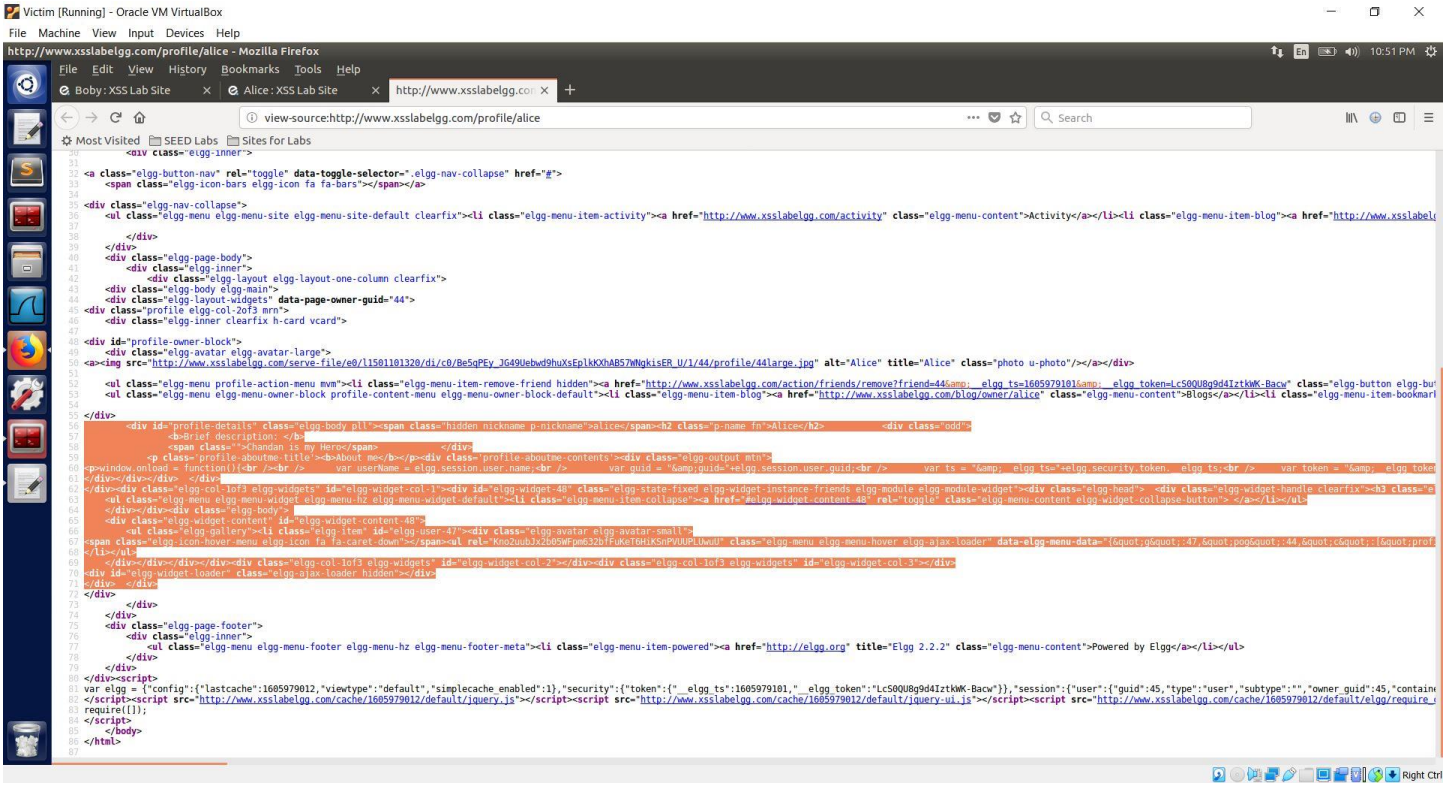


Now to test the application we will visit Alice's profile page who is a victim with the XSS worm from Boby's account. We can see that the entire Javascript code has been turned into text in the about me section and does not execute. This can be confirmed by checking the developer tools to check if a POST request has been sent to /action/profile/edit. This is shown in the screenshot below.





We can also view the page source to check why the code is not part of the script tag. From the below screenshot we can see that the new countermeasure removed the script tag and converted all special characters like “, ‘, & into their equivalent html encoding which has now made the Javascript code into a plain text. Hence, we can see that the countermeasure was really effective in mitigating XSS attack.


Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
http://www.xsslabelgg.com/profile/alice - Mozilla Firefox
10:51 PM
File Edit View History Bookmarks Tools Help
Boby : XSS Lab Site
Alice : XSS Lab Site
http://www.xsslabelgg.com
view-source:http://www.xsslabelgg.com/profile/alice
Search
Most Visited
SEED Labs
Sites for Labs
Right Ctrl

THANK YOU